STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*
Quarterly

Volume 32, Number 4B (101)

Grażyna ŚLUSARCZYK

PROJEKTOWANIE WIZUALNE Z WYKORZYSTANIEM HIERARCHICZNYCH HIPERGRAFÓW

Silesian University of Technology Press
Gliwice 2011

STUDIA INFORMATICA **Volume 32, Number 4B (101)**

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA* Nr kol. 1851

Editor in Chief

Editorial Board

STUDIA INFORMATICA is indexed in INSPEC/IEE (London, United Kingdom)

PL ISSN 0208-7286, QUARTERLY
Printed in Poland
The paper version is the original version

SPIS TREŚCI

CONTENTS

1. WPROWADZENIE

Komputerowe wspomaganie projektowania jest jednym z najistotniejszych zastosowań grafiki komputerowej. Programy, takie jak AutoCAD, Autodesk czy ArchiCAD, umożliwiają graficzne reprezentowanie modeli tworzonych obiektów i ułatwiają ich modyfikowanie. Rozwój technologii informatycznych, a szczególnie interaktywnych narzędzi komputerowych spowodował, że wspomaganie projektowania stało się jednym z zagadnień sztucznej inteligencji. Pojawiły się programy wspomagające twórczość inżynierską, architektoniczną (CAD) i artystyczną. Stworzone dotychczas systemy ułatwiające projektowanie wciąż jednak nie dostarczają wystarczającej informacji o strukturach modelowanych obiektów ani nie oferują odpowiedniego wsparcia na konceptualnym etapie projektowania, kiedy to są podejmowane najważniejsze decyzje projektowe.

Niezbędne stało się opracowanie formalnych modeli zarówno projektowanych obiektów, jak i procesu projektowego umożliwiających wspomaganie projektowania. Nastąpił szybki rozwój zarówno w dziedzinie dwu- a później trójwymiarowego modelowania, jak również reprezentacji i generacji projektów, tworzenia projektowych baz danych, symulacji i optymalizacji projektów oraz ich dokumentowania.

Rosnące obecnie zainteresowanie naukowców sposobem wykorzystania wiedzy i metodami działania projektantów ma na celu ulepszenie wspomagających ich narzędzi komputerowych, a jednocześnie przyczynia się do rozwoju nauki o projektowaniu. Lepsze zrozumienie zasad postrzegania informacji wizualnej i sposobu pracy projektanta ma prowadzić do znalezienia reprezentacji wewnętrznej obiektów, będącej uniwersalnym opisem ich struktur, umożliwiającym jednolite kodowanie wiedzy o projektowanych artefaktach.

1.1. Proces projektowy

Projektowanie jest procesem złożonym i trudnym do sprecyzowania, wymagającym świadomego wysiłku pozwalającego osiągnąć stan spełniający określone kryteria, i nieustannego podejmowania decyzji [Arch69]. Pomimo iż jest wyznaczony cel, do którego dążymy, nie można z góry określić sekwencji kroków prowadzących do rozwiązania problemu. Pierwsza

reprezentacja artefaktu powstaje w umyśle człowieka i jest poddawana operacjom mentalnym. Następnie dzięki wizualizacji, która stanowi przeniesienie myśli na szkice, rysunki, wykresy, mapy itp., powstają kolejne reprezentacje, a potem modele wyrażające idee projektanta.

Podczas procesu projektowego projektant wykonuje różne rodzaje akcji w celu zmiany reprezentacji w świecie zewnętrznym [SuGP00]. Akcje fizyczne dotyczą rysowania, kopiowania i usuwania elementów projektu. Akcje percepcyjne odnoszą się do postrzegania wizualnych cech przedstawionych elementów, takich jak kształty i rozmiary. Projektant zwraca także uwagę na relacje przestrzenne zachodzące pomiędzy tymi elementami, dostrzega przestrzenie istniejące pomiędzy narysowanymi elementami i odkrywa zależności między nimi. Akcje funkcyjne przypisują cechom wizualno-przestrzennym obiektów znaczenie, funkcje lub pojęcia abstrakcyjne. Akcje koncepcyjne polegają na określaniu celów i wymagań projektowych i pozwalają projektantowi oceniać otrzymane rozwiązania [SuGP00].

Ważną cechą projektowania jest jego dynamizm. Przestrzeń projektowa, w której zachodzi proces projektowy, składa się z definicji rozważanego problemu, kryteriów projektowych oraz przestrzeni dopuszczalnych rozwiązań projektowych. Proces projektowania może być więc rozumiany jako poszukiwanie ścieżek w przestrzeni projektowej prowadzących do potencjalnych rozwiązań, które jest sterowane za pomocą założeń projektowych opartych na aktualnych wymaganiach i ograniczeniach. Jednak nie wszystkie wymagania są już znane na początku procesu projektowego, niektóre z nich pojawiają się dopiero w trakcie jego trwania. Interakcja człowieka z reprezentacją zewnętrzną będąca wnioskowaniem wizualno-przestrzennym nie tylko pozwala łączyć formę artefaktu z jego funkcją, ale także wpływa na reprezentację wewnętrzną istniejącą w umyśle i dokonywanie zmian w zewnętrznej reprezentacji tworzonego projektu. Wykonywanie akcji zmieniających stan projektu (rysowanie, kopiowanie, usuwanie), percepcja cech i relacji pomiędzy przedstawionymi elementami oraz ocena rozwiązań częściowych powodują pojawianie się nowych koncepcji i wymagań. Z każdą nową koncepcją lub wymaganiem poszerza się przestrzeń projektowa. Z kolei nowe koncepcje powodują konieczność modyfikacji generowanych rozwiązań, co poszerza przestrzeń rozwiązań, będącą zbiorem stanów reprezentujących potencjalne rozwiązania projektowe. W ten sposób przestrzeń projektowa i przestrzeń rozwiązań ewoluują wspólnie wraz z pojawianiem się w procesie projektowym nowych wymagań i rozwiązań.

Cele, wymagania i ograniczenia nie są jedynymi czynnikami wpływającymi na postać projektu. Projekt często powinien też być zgodny z pewnym systemem konwencji, nazywanym stylem. Projektant wykorzystuje dostępną mu wiedzę projektową, aby otrzymać model obiektu posiadający cechy i właściwości odzwierciedlające jego intencje. Zmieniający się kontekst projektowy zawierający zarówno cele i ograniczenia projektowe, jak również intencje projektanta istotnie wpływa na rozwiązywane zadanie projektowe [Knig04]. Mając na

uwadze wiedzę projektową, specyfikację funkcjonalną projektowanego obiektu, pożądane cele, wymagania i narzucone ograniczenia, projektant tworzy opis obiektu zgodnego z określoną specyfikacją [Topp98].

Projektanci operują na różnych poziomach abstrakcji. Najczęściej projektowanie rozpoczyna się od ogólnej koncepcji, a następnie dodaje do projektu bardziej szczegółowe elementy (projektowanie metodą top-down). Wynik procesu projektowego na wyższym poziomie steruje kolejnymi krokami procesu odbywającymi się na niższych poziomach abstrakcji. Odpowiednio w procesie projektowym wspomaganym narzędziami komputerowymi (CAD) wyróżnia się trzy fazy: projektowanie koncepcyjne, projektowanie szczegółowe i dokumentowanie projektu.

Głównym celem projektowania koncepcyjnego jest określenie wymagań funkcjonalnych i ograniczeń, będących wynikiem rozmowy z klientem, oraz stworzenie prototypowego projektu spełniającego wyspecyfikowane kryteria. Zwykle projektant na podstawie ogólnej koncepcji projektu szkicuje najpierw prototypowe rozwiązanie wysokiego poziomu bez zagłębiania się w szczegóły. Rozwiązanie takie ułatwia porozumienie z klientem i jest podstawą do dalszej dyskusji na temat projektu. Uzgodnienie intencji i preferencji klienta może spowodować konieczność modyfikacji wymagań i/lub rozwiązania prototypowego. Dopiero w kolejnym kroku projektant rozpoczyna opracowanie szczegółowego projektu. Projektowanie szczegółowe polega na udoskonaleniu schematycznego projektu stworzonego w fazie koncepcyjnej i dodaniu do niego szczegółowych informacji.

Komputerowe narzędzia wspomagające projektowanie stają się coraz popularniejsze wśród projektantów, gdyż znacznie ułatwiają ich pracę. W przypadku dwuwymiarowych narzędzi CAD-owskich projektowanie odbywa się w sposób podobny jak na desce kreślarskiej, jednak modyfikacje projektu są znacznie łatwiejsze. W przypadku wspomagania projektowania w trzech wymiarach, z trójwymiarowych obiektów geometrycznych można z łatwością uzyskać informacje dotyczące ich objętości i powierzchni, co znacznie ułatwia analizę projektu.

Narzędzia CAD-owskie można podzielić na narzędzia ogólne i wyspecjalizowane. Narzędzia ogólne wspomagają szeroką grupę projektantów różnych specjalności (szczególnie architektów i inżynierów). Zawierają one tylko podstawowe geometryczne elementy projektów, jak np. odcinki i okręgi, i nie wspomagają projektowania elementów specyficznych dla danej dziedziny, jak ściany budynków. Projektowanie takich elementów wspomagają natomiast wyspecjalizowane narzędzia przeznaczone dla określonej grupy projektantów. Stanowią one zwykle nadbudowę jakiegoś ogólnego narzędzia projektowego. Przykładami wyspecjalizowanych narzędzi CAD-owskich przeznaczonych dla architektów są Architectural Desktop [Auto02], ArchiCAD [ArCA10] i Autodesk Revit [AuRA10].

Obecnie standardem projektowym dla narzędzi CAD-owskich przeznaczonych dla architektów, inżynierów i konstruktorów, a więc dedykowanych dziedzinie AEC (Architecture, Engineering & Construction), staje się BIM (Building Information Modelling) [EaTe08]. Narzędzia, takie jak Allplan [Allp10], ArchiCAD [ArCA10] czy Autodesk Revit [AuRA10] deklarują swoją zgodność z modelem IFC (Industry Foundation Classes) [InFC08], który jest popularnym formatem dla BIM, zapewniającym wymianę danych pomiędzy aplikacjami dotyczącymi różnych dziedzin. Istotą metodologii BIM jest modelowanie wykorzystujące sparametryzowane obiekty [LeSe06], gdzie komponenty projektowanych budynków są opisywane jako hierarchicznie zagnieżdżone obiekty posiadające atrybuty i reguły określające ich zastosowanie. BIM umożliwia generowanie danych dotyczących budynku i zarządzanie nimi w trakcie jego istnienia. Dane te obejmują geometrię budynku, relacje przestrzenne, informacje geograficzne oraz liczbę i własności elementów składowych budynku. Niestety, dostępne systemy BIM wciąż nie posiadają zadowalających możliwości konfigurowania obszarów przestrzennych.

Stworzone dotychczas komercyjne narzędzia CAD-owskie nie wspomagają koncepcyjnej fazy projektowania w dostatecznym stopniu. Nie wspierają także projektantów w trakcie całego procesu projektowego w podejmowaniu decyzji projektowych, które prowadziłyby do uzyskania interesujących, oryginalnych i innowacyjnych rozwiązań. Wspomagają one jedynie projektowanie rutynowe, polegające na generacji wszystkich możliwych rozwiązań spełniających zadane na początku ograniczenia i zakładające niezmienność przestrzeni projektowej bez uwzględnienia dynamicznie zmieniającego się kontekstu projektowego. Komputerowe systemy wspomagające dynamiczne projektowanie pozostają wciąż domeną badań uniwersyteckich [Yake00, Maed03].

1.2. Wizualizacja w projektowaniu

We współczesnym projektowaniu wspomaganym komputerowo wzrasta rola wizualizacji na wszystkich etapach procesu projektowego. W projektowaniu wizualnym różnego rodzaju obiekty wizualne są wykorzystywane do reprezentacji, poszukiwania i modyfikacji rozwiązań zadań projektowych. Dotyczy to różnych dziedzin projektowania, zarówno projektowania architektonicznego, przemysłowego, graficznego, jak i artystycznego [Grab07]. Projektowanie wizualne wspomaga systematyczne myślenie projektanta, wizualną percepcję cech projektowanych obiektów i odkrywanie nowych wymagań projektowych. Emergencja nowych idei koncepcyjnych bierze swoje źródło właśnie w wizualnej reprezentacji, wizualnym postrzeganiu i wnioskowaniu [SuGP00]. Wizualna ocena projektowanego obiektu ułatwia projektantowi dokonanie wyboru najlepszego rozwiązania. Dlatego też odpowiednia wizualnizacja projektowanych obiektów, a także procesu projektowego odgrywa znaczącą rolę w podejmowaniu przez projektanta istotnych decyzji projektowych.

W projektowaniu wizualnym często są stosowane metody lingwistyczne, wykorzystujące podobieństwo pomiędzy procesem projektowania a sposobem tworzenia zdań w języku naturalnym. Formalne modele metod lingwistycznych wykorzystują koncepcje gramatyk, będących zbiorami reguł pozwalających odpowiednio rozmieszczać elementy obiektów. Wiedza syntaktyczna zakodowana w regułach gramatyk pozwala generować opisy projektów, zgodne z narzuconymi wymaganiami. Natomiast wiedza semantyczna pozwala dokonać odpowiedniej interpretacji opisów projektów. Zbiór obiektów wizualnych, będących zinterpretowanymi opisami projektów utworzonymi na podstawie wiedzy syntaktycznej i semantycznej dla danej dziedziny, jest nazywany językiem wizualnym. Gramatyka stanowi więc narzędzie projektowe, umożliwiające generację rozwiązań projektowych w postaci elementów języka wizualnego.

Popularnym typem języków wizualnych są diagramy, będące skończonymi konfiguracjami elementarnych kształtów geometrycznych. Projektowanie wizualne, polegające na wykonywaniu akcji projektowych na diagramach opisujących projekty, nazywane jest projektowaniem diagramowym [Grab03, SuGP00]. W [ChHY97] są rozważane własności diagramów stosowanych w procesie projektowania wizualnego. Każdy diagram jest traktowany jako fragment programu, a transformacje wykonywane na skutek interakcji użytkownika z diagramem są rozumiane jako programowanie. Taka charakteryzacja wizualnego procesu projektowego pozwala zastosować do niego zasady wykorzystywane w inżynierii oprogramowania, prowadząc do wizualnego sposobu tworzenia oprogramowania.

Mimo iż w wielu dziedzinach nauki i inżynierii projektanci posługują się wizualnymi diagramami podczas modelowania i projektowania obiektów, to istniejące narzędzia CAD-owskie służące tworzeniu i przedstawianiu na ekranie diagramów wizualnych nie są w stanie adekwatnie wspomagać projektanta w wizualizacji i innowacyjnym projektowaniu z użyciem diagramów, gdyż nie dają możliwości efektywnego przedstawiania tych samych diagramów na różnych poziomach abstrakcji. Niniejsza rozprawa jest jedną z prób poprawienia tej sytuacji.

1.3. Wewnętrzna reprezentacja obiektów wizualnych

Istotnym problemem w projektowaniu jest możliwość automatycznego przetwarzania wewnętrznej reprezentacji obiektów wizualnych odpowiadających modelom projektowanych artefaktów.

Często stosowaną przez projektantów reprezentację wewnętrzną obiektów stanowią różnego rodzaju grafy. Pojawiają się one głównie w koncepcyjnej fazie projektowania, kiedy projektanci operują na wysokim poziomie abstrakcji, opisując funkcjonalność i strukturę projektowanych obiektów [Kraf03, KrNa07]. Informacja o strukturze obiektu jest reprezentowa-

na poprzez etykiety wierzchołków i krawędzi grafu oraz przez samą strukturę grafu odzwierciedlającą relacje między elementami obiektu. Większość projektowanych obiektów posiada strukturę hierarchiczną, tzn. składają się one z wielu komponentów, które z kolei zawierają następne części składowe. Struktura taka często jest reprezentowana za pomocą hierarchicznych grafów, które pozwalają wyrażać skomplikowane relacje, zachodzące pomiędzy częściami składowymi złożonych obiektów. Umożliwiają też reprezentację obiektów na różnych poziomach szczegółowości [DrHP00, GrŚP03]. Grafy hierarchiczne znalazły więc zastosowanie nie tylko w projektowaniu architektonicznym i inżynierskim, ale także w modelowaniu procesów biznesowych, modelowaniu oprogramowania itd.

Reprezentacje grafowe projektowanych obiektów mogą być tworzone i modyfikowane za pomocą reguł przepisywania grafów, definiujących sposób transformacji grafu. Reguła przepisywania składa się z grafu lewej strony i grafu prawej strony. Aby reguła mogła być zastosowana, graf izomorficzny z grafem lewej strony reguły musi zostać znaleziony w generowanym grafie. Graf ten jest następnie zastępowany grafem izomorficznym z grafem prawej strony reguły. Reguły przepisywania grafów wraz z grafem początkowym tworzą gramatykę grafową, będącą narzędziem generacyjnym definiującym język grafowy złożony z grafów wygenerowanych przez tę gramatykę.

Pojęcie gramatyki grafowej i przepisywania grafów zostało wprowadzone przez J. Pfaltza [PfRo69] i H. J. Schneidera [Schn70]. Zostało zainspirowane przez gramatyki Chomsky'ego [Chom65], przepisywanie termów [BaNi99] i sieci Petriego [Petr62]. Gramatyki grafowe znalazły zastosowanie w wielu dziedzinach informatyki, takich jak reprezentacja, rozpoznawanie, generowanie i przetwarzanie obrazów (wzorców) [Flas89, Flas07, FuKS74, GoTh78, Pavl77, Rose76, Shaw69, TaFl91], semantyka języków programowania [Gott83, Prat71, Nagl79, Pada82], definiowanie języków wizualnych [EEKR99], sterowanie procesami [MaWi82], specyfikacja typów danych [Schn70, Ehri83], konstrukcja kompilatorów [Hoff83, Schn75], inżynieria oprogramowania [LeNa84], modelowanie systemów współbieżnych i rozproszonych [B0FH85, CaMo83] oraz projektowanie baz danych [Bart79, Furt79, Schn79, EhKr80, Meie83].

W literaturze były też rozważane zagadnienia, dotyczące wykorzystania różnych rodzajów gramatyk grafowych w projektowaniu graficznym. Teoretyczne podstawy modelowania graficznego z użyciem kompozycyjnych gramatyk grafowych są przedstawione w [Grab93], a zastosowanie gramatyk grafowych do wspomagania koncepcyjnej fazy projektowania inżynierskiego jest opisane w [KrMN02]. Wykorzystanie gramatyk grafowych w generacyjnym podejściu do wspomagania projektowania inżynierskiego jest przedstawione między innymi w [Flas95], gdzie gramatyki grafowe NLC służą do opisu części mechanicznych, w [BoGr95] i [BGNS03], gdzie kompozycyjne gramatyki grafowe zostały zastosowane do projektowania mostów i struktur szkieletowych, a także w [Ślus03], gdzie hierarchiczne gramatyki hiper-

grafowe generują reprezentacje obwodów elektrycznych. Było również rozważane zastosowanie gramatyk grafowych na koncepcyjnym etapie projektowania architektonicznego [GoGN90, BoSz01, SzSB02, Szub05], zastosowanie kompozycyjnych gramatyk grafowych w projektowaniu architektonicznym do generowania rozkładów pomieszczeń [GrBo97] i ich aranżacji [Ślus04], w projektowaniu przemysłowym (ekspresy do kawy) i artystycznym (grafiki w stylu Eschera) [Ślus99]. Wspomniane tu badania nad problematyką projektowania i modelowania graficznego miały istotny wpływ na stworzenie przedstawionego w rozprawie języka wizualnego.

Jedną z najbardziej obiecujących reprezentacji projektowanych obiektów stanowią hipergrafy. Ich podstawowymi elementami są etykietowane hiperkrawędzie, które umożliwiają reprezentowanie relacji wieloargumentowych. Hipergraf jest tworzony przez połączenie danego zbioru wierzchołków za pomocą hiperkrawędzi. Każda hiperkrawędź posiada uporządkowane zbiory końcówek (ang. tentacles) wejściowych i wyjściowych przyczepionych do wierzchołków. Różne końcówki jednej hiperkrawędzi mogą być przyczepione do tego samego wierzchołka. W [HaKr87] zostały opisane hipergrafy i ich transformowanie za pomocą zastępowania hiperkrawędzi, podczas gdy przepisywanie hierarchicznych hipergrafów jest rozważane w [DrHP00].

Struktury grafowe mogą być także transformowane z wykorzystaniem operacji grafowych [BoGH99, Ślus03]. Podczas gdy reguły przepisywania grafów w gramatykach są stosowane głównie w celu lokalnych modyfikacji grafów, operacje grafowe pozwalają na dokonywanie znacznych zmian w grafach, np. poprzez dołączanie do nich nowych fragmentów lub usuwanie wybranych podstruktur. Operacje grafowe umożliwiają tworzenie i modyfikowanie grafów oraz zmianę poziomu szczegółowości reprezentacji, dzięki czemu ułatwiają przeszukiwanie przestrzeni rozwiązań poprzez wspomaganie wnioskowania dotyczącego projektów.

Wyróżnia się trzy główne podejścia do opisu grafów i transformacji grafowych: z wykorzystaniem teorii zbiorów [Nagl73, Nagl79, Nagl87], teorii kategorii [HeSt73, EhLo93, Cour88], logiki [BaCo86, BaCo87, Schn91].

Powstało wiele narzędzi umożliwiających modelowanie i implementację transformacji grafowych. Należy tu wspomnieć o narzędziach ogólnego przeznaczenia, takich jak PROGRES [PROG08], FUJABA [Fuja10] czy AGG [AtGG10] oraz bardziej specjalistycznych narzędziach, jak GENGED [BaEh99] – system generujący edytory języków wizualnych, DIAGEN [MiVi95] – system generujący edytory diagramów, MetaEnv – silnik odwzorowujący notację diagramową na sieci Petriego.

Systemy wspomagające projektowanie, które ilustrują rozważania niniejszej rozprawy, zawierają implementacje transformacji hipergrafów, będących wewnętrzną reprezentacją projektowanych obiektów.

1.4. Reprezentacja wiedzy projektowej

Aby narzędzia komputerowe mogły efektywnie wspierać projektowanie, a szczególnie jego etap koncepcyjny, reprezentacja wewnętrzna obiektów powinna odzwierciedlać nie tylko ich strukturę, ale także umożliwiać kodowanie wiedzy projektowej. Obecnie łączy się narzędzia CAD-owskie ułatwiające projektantowi wykonywanie operacji projektowych z systemami ekspertowymi, w których zaszyta jest wiedza z danej dziedziny projektowej. Wciąż jednak tworzenie oprogramowania wspomagającego pracę koncepcyjną projektanta w widoczny sposób napotyka barierę braku spójności pomiędzy wiedzą z danej dziedziny (architektura, budownictwo, sztuka użytkowa) a jej wewnętrzną reprezentacją w programie komputerowym.

Wiedzę można tu rozumieć jako stwierdzenia dotyczące relacji i logicznych powiązań między faktami. Relacje te pozwalają nam wnioskować o nowych faktach wynikających z tych już istniejących. Problem reprezentacji wiedzy polega na poszukiwaniu najlepszej możliwej struktury danych, która przechowywałaby wiedzę z danej dziedziny w sposób umożliwiający efektywne jej wykorzystanie podczas rozwiązywania problemów projektowych. Wybór odpowiedniej metody reprezentacji wiedzy jest często kluczowy dla opracowania, funkcjonalności i rozszerzeń systemu komputerowego wspomagającego użytkownika.

Istnieje wiele różnych metod reprezentacji wiedzy, z których każda ma swoje zalety i nadaje się szczególnie dobrze do pewnych określonych zadań [Szub05]. Najpopularniejsze z tych metod to:

- metody wykorzystujące logikę [Gara84] – szczególnie użyteczna do definiowania ograniczeń i reguł projektowych,
- sieci semantyczne [Quil68] – odpowiednia do definiowania relacji koncepcyjnych,
- kadry (ang. frames) [Mins75] – wspomaga modularyzację i wielokrotne wykorzystywanie pojęć,
- metody wykorzystujące reguły [HRWL83, Nebe88] – odpowiednie do definiowania reguł projektowych i manipulowania projektowanymi obiektami,
- metody zorientowane obiektowo [RBPE97, Sigf96] – wspomagają modularyzację,
- metody hybrydowe łączące różne metody reprezentacji [PROG08].

Wymienione powyżej metody są nazywane symbolicznymi reprezentacjami wiedzy. Wiedza jest tu reprezentowana explicite w postaci terminów symbolicznych, a wnioskowanie polega na manipulacji tymi terminami. Reprezentacje symboliczne są wykorzystywane w systemach komputerowych opartych na wiedzy (m.in. systemy ekspertowe). Innym sposobem reprezentacji wiedzy jest reprezentacja, w której organizacja, przetwarzanie i manipulowanie wiedzą przebiega w sposób wzorowany na organizmach żywych. Do tego

rodzaju reprezentacji zaliczamy sieci neuronowe [Lipm87, Wass89] i algorytmy genetyczne [Gold89].

Wyróżnia się dwa rodzaje symbolicznej reprezentacji wiedzy:

- proceduralny – zawiera procedury definiujące sposób wykonywania pewnych zadań i rozwiązywania danych problemów z wykorzystaniem wiedzy z określonej dziedziny,
- deklaratywny – zawiera zbiór faktów i reguł specyficznych dla określonej dziedziny.

Jedną z istotnych metod reprezentacji wiedzy jest zapoczątkowana przez Chomsky'ego metoda lingwistyczna, gdzie są wykorzystywane różnego rodzaju gramatyki (ciągowe, grafowe, gramatyki kształtu). W świetle powyższych rozważań metoda ta może zostać scharakteryzowana jako deklaratywna metoda symboliczna wykorzystująca reguły. W metodzie lingwistycznej podstawowe elementy obiektu, zwane prymitywami, są traktowane jak litery alfabetu, a skomplikowane obiekty – jak złożenie słów zdefiniowanych nad tym alfabetem. Reguły rządzące składaniem prymitywów w obiekt definiują gramatykę określonego języka. Elementy języka generowane przez daną gramatykę są więc strukturalnym opisem klasy dopuszczalnych obiektów w postaci zbiorów prymitywów i operacji odpowiedniego ich składania.

W projektowaniu architektonicznym często są wykorzystywane gramatyki kształtu [StGi72, Gips99], które pozwalają projektantowi uchwycić cechy charakterystyczne dla pewnego stylu (projektowanie domów w stylu Franka Lloyda Wrighta [KoEi81], królowej Anny [Flem87], willi Palladiańskich [StMi78], krzeseł w stylu Hepplewhite [Knig80]). Moc opisowa tych gramatyk jest jednak ograniczona, gdyż ich prymitywy są czysto geometrycznymi dwu- lub trójwymiarowymi kształtami. Znacznie korzystniejsze wydaje się użycie w systemach CAD gramatyk grafowych, gdyż generowane przez nie grafy pozwalają zakodować znacznie większą ilość informacji niż ciągi czy kształty. Gramatyki grafowe umożliwiają modelowanie złożonych transformacji grafowych i specyfikowanie nieskończonej klasy grafów o takiej samej strukturze. W [Grab94] dla celów projektowych została wprowadzona specjalna postać grafowej reprezentacji obiektów generowanej przez gramatyki grafowe. Struktura projektowanego obiektu jest tu reprezentowana za pomocą grafu kompozycyjnego, którego wierzchołki odpowiadają komponentom modelu, a krawędzie reprezentują relacje między nimi. Wykorzystanie operacji na grafach kompozycyjnych i ich transformacji za pomocą atrybutowanych kompozycyjnych gramatyk grafowych w projektowaniu graficznym zostało przedstawione w [Ślus99].

Aby komputerowy system wspomagania projektowania wykorzystujący wiedzę mógł pomagać użytkownikowi projektować lepiej i szybciej, powinien oprócz bazy wiedzy (przechowującej fakty dotyczące danej dziedziny), dynamicznego kontekstu (zawierającego wymagania projektowe, parametry aktualnie rozwiązywanego problemu, rozwiązania pośrednie

i informacje generowane przez system w trakcie działania) oraz interfejsu użytkownika (umożliwiającego definiowanie problemów i odczytywanie otrzymanych rozwiązań), posiadać także mechanizm wnioskowania, który ostrzega użytkownika przed tworzeniem niepoprawnych rozwiązań oraz pozwala modyfikować i rozszerzać kontekst projektowy. System taki powinien wspomagać koncepcyjną fazę projektowania poprzez automatyczne podejmowanie pewnych decyzji, poszukiwanie rozwiązań problemów podobnych do rozważanego wśród rozwiązań już istniejących (ang. case-based reasoning), adaptację i/lub optymalizację znalezionych rozwiązań, a także sprawdzanie kompatybilności rozwiązań z normami i standardami projektowymi.

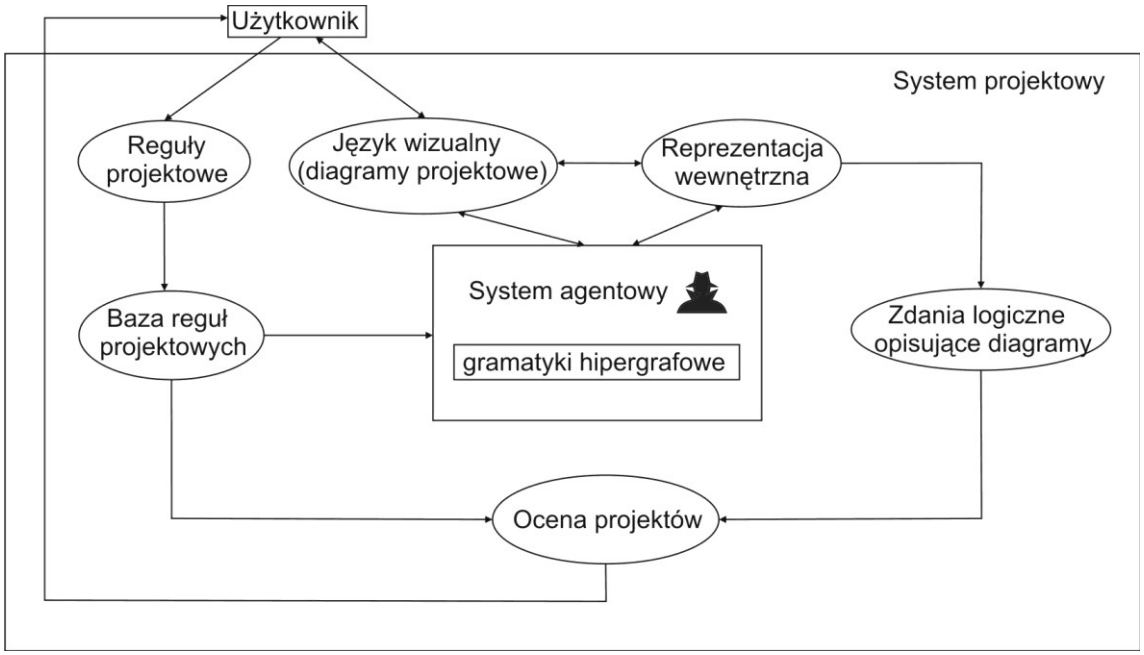
Istniejące systemy wspomagania projektowania, takie jak na przykład SEED [FlWo95] (system wspomagający koncepcyjną fazę projektowania architektonicznego), CADET [CADE11] (system wspomagający projektowanie schematów mechanicznych), FABEL [VoBA97] (system wspomagający projektowanie architektury przemysłowej), IPA [Poko03] (system pozwalający przechowywać główne kroki procesu projektowego dla danego problemu) nie są w wystarczającym stopniu przydatne w koncepcyjnej fazie projektowania. W żadnym z nich nie zostały też wykorzystane transformacje grafowe.

Rozprawa ta jest więc próbą stworzenia formalnego modelu systemu wspomagania projektowania, który, wykorzystując wiedzę projektową, umożliwiałby efektywne otrzymywanie poprawnych rozwiązań projektowych i spełniał wymienione powyżej wymagania.

1.5. Cel i układ pracy

W prezentowanej pracy są przedstawione teoretyczne podstawy konstrukcji wizualnego systemu projektowego, wykorzystującego wiedzę projektową, i wspomagającego projektanta na etapie projektowania koncepcyjnego. Idea takiego systemu powstała w trakcie prac prowadzonych w ramach grantu KBN pt. ”*Transformacje grafowe w konstrukcji systemów wnioskowania diagramowego*” w latach 2003-2006 [Spra06]. System ten posiada zorientowany problemowo edytor, pozwalający tworzyć rysunki reprezentujące rozwiązania rozważanego problemu projektowego. Użytkownik porozumiewa się z systemem za pomocą języka wizualnego opartego na konceptualizacji projektowej specyfikującej pojęcia wykorzystywane w tym języku. Projektant ma również możliwość wprowadzania reguł wyrażających wymagania projektowe, które będą automatycznie sprawdzane przez system w trakcie zachodzenia całego procesu projektowego. Tworzone przez użytkownika rysunki są automatycznie przekształcane na wewnętrzną reprezentację, która może być efektywnie modyfikowana podczas wykonywania akcji projektowych. Reprezentacja ta jest transformowana do postaci zdań logicznych opisujących generowane rozwiązania. Moduł wnioskowania sprawdza poprawność tych zdań względem formuł logicznych, reprezentujących ograniczenia projektowe oraz wprowadzone wcześniej wymagania projektowe.

Istotnym elementem systemu wspomagania projektowania wizualnego jest zaproponowany tu system agentów projektowych, będących inteligentnymi asystentami wspomagającymi projektanta w podejmowaniu decyzji. Zachowanie agentów jest modelowane za pomocą reguł gramatyk, umożliwiających generowanie wewnętrznych reprezentacji potencjalnych rozwiązań zadań projektowych. Agenci oceniają zgodność tych reprezentacji względem predykatów opisujących kryteria projektowe. Poprawne rozwiązania są przedstawiane w postaci elementów języka wizualnego, które mogą być następnie modyfikowane przez projektanta za pomocą akcji projektowych. Ogólna struktura systemu wspomagania projektowania wizualnego z wykorzystaniem agentów projektowych jest przedstawiona na rys. 1.1.



Rys. 1.1. Struktura systemu wspomagania projektowania wizualnego
Fig. 1.1. The structure of a visual design supporting system

Omówiony jest tu abstrakcyjny diagramowy język projektowania niezależny od konkretnej dziedziny projektowej. Język ten bazuje na zaproponowanej w pracy konceptualizacji dziedziny projektowania wizualnego, która definiuje podstawowe pojęcia z tej dziedziny wraz z ich własnościami oraz zachodzące między nimi relacje. Jednoznaczność przekazu wiedzy na temat projektowania zapewnia kategoryzacja pojęć pozwalająca przypisywać obiekty do określonych klas, wraz z taksonomią umiejscawiającą klasy obiektów w hierarchicznej strukturze. Konceptualizacja projektowa umożliwia jednoznaczne używanie wizualnego języka projektowania do rozważania problemów projektowych i służy jako podstawa do zdefiniowania mechanizmu wnioskowania logicznego o właściwościach projektów złożonych z wyspecyfikowanych w konceptualizacji pojęć.

Diagramowy język projektowania jest zbiorem uogólnionych diagramów projektowych, które zawierają informację o projektowanym obiekcie zredukowaną do podstawowych składowych wizualnych wyrażających najistotniejsze cechy niezbędne do opisu znaczenia [Grab07]. Każdy element diagramowego języka wizualnego reprezentuje całą klasę rozwiązań projektowych spełniających określone wymagania.

Ponieważ elementy diagramowego języka projektowania dla każdej dziedziny zastosowań są definiowane w inny sposób, język ten może być traktowany jako rodzina języków projektowych, gdzie są określone wspólne dla tych języków ogólne reguły tworzenia diagramów projektowych. Przypisując elementom konceptualizacji projektowej obiekty i relacje specyficzne dla danej dziedziny projektowania, otrzymuje się konceptualizację zadaniową [Guar97], a wykorzystujący ją język diagramowy staje się zorientowany problemowo, jak np. język projektowania związków chemicznych, kratownic, mebli, budynków czy rozkładów ich pomieszczeń.

Omawiane narzędzie w postaci diagramowego języka wizualnego pośredniczącego pomiędzy wiedzą z danej dziedziny a jej wewnętrzną reprezentacją w programie komputerowym może być wykorzystywane w różnych typach projektowania. Użytkownik systemu wspomagającego projektowanie operuje na przyjaznej dla niego reprezentacji graficznej w postaci diagramów. Przedstawiony język opisu projektów wykorzystuje teorię hipergrafów. Wiedza syntaktyczna i semantyczna dotycząca rozwiązywanego problemu i zawarta w diagramach jest automatycznie kodowana w postaci atrybutowanych hierarchicznych hipergrafów. Do transformowania takiej reprezentacji wewnętrznej są wykorzystywane zarówno zdefiniowane tu hierarchiczne gramatyki hipergrafowe będące narzędziem generacyjnym umożliwiającym tworzenie projektów zgodnych z narzuconymi regułami, jak i wyspecyfikowane operacje na hierarchicznych hipergrafach, pozwalające na efektywne dokonywanie modyfikacji odpowiadających akcjom projektowym i uwzględniające dynamicznie zmieniający się kontekst projektowy.

Wiedza zawarta w hipergrafowej reprezentacji projektów jest automatycznie tłumaczona na formuły logiczne opisujące projekty. Zastosowane w pracy ontologiczne podejście do modelowania komputerowego wspomagania projektowania pozwala sformalizować reprezentację wiedzy z dziedziny projektowej za pomocą konceptualizacji projektowej i języka logicznego, będącego zbiorem formuł wyrażających własności diagramów projektowych. Zdefiniowane tu ontologiczne dopasowanie symboli słownika formuł do pojęć konceptualizacji projektowej umożliwia wykorzystanie logiki pierwszego rzędu do wyrażania wiedzy dotyczącej obiektów projektowych reprezentowanych przez komponenty diagramów. Interpretacja formuł logicznych jest określana za pomocą struktury relacyjnej przypisującej atomy hipergrafów i wartości ich atrybutów elementom składowym formuł. Taka reprezentacja wiedzy pozwala w efektywny sposób wnioskować na temat tworzonych projektów. Ponieważ

ograniczenia i wymagania projektowe również mogą być wyrażane w postaci formuł logicznych pierwszego rzędu, bazujących na takiej samej konceptualizacji projektowej, system wspomagający projektowanie może sprawdzać zachodzenie żądanych kryteriów projektowych poprzez porównywanie określonych zbiorów formuł logicznych. Formalizacja wiedzy o projektach zawartej w hipergrafach może być także wykorzystana w modelu IFC, będącym formatem dla BIM. Model IFC może zostać następnie użyty do sprawdzenia zgodności projektu z wymaganiami nieuwzględnionymi na etapie projektowania koncepcyjnego [Yurc09].

Przedstawiona tu formalna metoda łącząca wizualną reprezentację projektów wykorzystującą język diagramowy z ich wewnętrzną reprezentacją w postaci atrybutowanych hierarchicznych hipergrafów stanowi model formalny dla zaproponowanego w pracy systemu agentów projektowych wspomagających projektowanie kreatywne poprzez generowanie wielu alternatywnych rozwiązań zadania projektowego. Konceptualizacja projektowa służy tu do komunikacji pomiędzy agentami i umożliwia odpowiednią interpretację otrzymywanych przez nich informacji. Natomiast logiczny model wnioskowania umożliwia agentom przetwarzanie informacji i sprawdzanie zgodności generowanych rozwiązań z zadanymi kryteriami projektowymi.

Za oryginalny wkład do rozwoju dziedziny projektowania wizualnego Autorka uważa zastosowanie ontologicznego podejścia do reprezentacji wiedzy projektowej, gdzie konceptualizacja projektowa umożliwia opracowanie języka wizualnego służącego do celów projektowania koncepcyjnego. Nowym pomysłem są odwzorowania struktur projektów w dziedzinę diagramów projektowych pozwalające otrzymać wizualizacje projektów w postaci diagramów będących elementami tego języka. Autorka brała udział w specyfikacji pojęcia atrybutowanych hierarchicznych hipergrafów rozmieszczenia stanowiących reprezentację wewnętrzną diagramów projektowych [GLŁŚ07]. Natomiast definicje operacji na stosowanych tu hierarchicznych hipergrafach i generujących te hipergrafy hierarchicznych gramatyk rozmieszczenia są oryginalnym wkładem Autorki przedstawionym w tej pracy. Autorka brała również udział w opracowaniu logicznego modelu wnioskowania opartego na formułach pierwszego rzędu, które wyrażają wiedzę projektową dzięki zastosowaniu do ich interpretacji struktury relacyjnej [GrŚl11]. Ten model wnioskowania został wykorzystany w implementacji prototypowego systemu wspomagającego projektowanie i wnioskowanie *HSSDR* wykonanej przez Szymona Gajka [Gaje11].

Oryginalną częścią pracy jest też zaproponowany system agentów projektowych, których zachowanie jest modelowane za pomocą reguł hierarchicznych gramatyk rozmieszczenia generujących wewnętrzne reprezentacje potencjalnych rozwiązań zadań projektowych i oceny tych rozwiązań wykorzystującej modalny rachunek predykatów. Autorka opracowała teoretyczne podstawy prototypowego wieloagentowego systemu aranżacji wnętrz zaimplemento-

wanego przez Tomasza Zarasia [Zara09], który to system zapewnia modularność i współbieżne przetwarzanie informacji dotyczących różnych fragmentów projektu.

W dalszej części tego podrozdziału zostanie krótko scharakteryzowana treść kolejnych rozdziałów pracy.

Rozdział drugi omawia ogólną charakterystykę języków wizualnych oraz ich rolę w projektowaniu, a także wprowadza diagramy projektowe. Następnie jest przedstawiona syntaktyka i semantyka projektowo zorientowanego języka wizualnego, którego zdania mają postać diagramów projektowych opisujących wykreowane rozwiązania zadań projektowych. W procesie wizualnego projektowania diagramy są przekształcane w sposób umożliwiający reprezentowanie projektowanego obiektu na różnych poziomach szczegółowości. Projektant rysuje początkowy diagram projektowy, a następnie uszczegółowia go aż do osiągnięcia opisu końcowego projektu. Diagramy pojawiające się na kolejnych etapach projektowania stanowią konfigurację projektową, będącą sekwencją zdań projektowego języka wizualnego.

Każdy element języka wizualnego możne być traktowany jako uogólniony diagram reprezentujący pewną klasę rozwiązań projektowych, spełniających określone wymagania. Nie stanowi on jednak ukończonego rozwiązania projektu, które jest wykorzystywane podczas wizualizacji projektu w narzędziach CAD-owskich. Diagramy projektowe zawierające ogólne idee o projektowanych obiektach pełnią rolę podobną do szkiców projektowych stosowanych przez projektanta w koncepcyjnej fazie projektowania, a język diagramowy umożliwia projektantowi śledzenie zmian dokonywanych w tych diagramach. Wykorzystanie diagramów projektowych jako formalnego języka reprezentacji wiedzy pozwala zdefiniować komputerowo wspomagane metody syntezy obrazów oraz wnioskowania z obrazów opartego na odpowiedniej ich analizie.

W rozdziale tym została zdefiniowana konceptualizacja dziedziny projektowania wizualnego, będąca specyfikacją pojęć z tej dziedziny, ich taksonomii oraz zachodzących między nimi relacji. Zostały przedstawione także reguły przekształcania struktur projektowych złożonych z obiektów określonych poprzez konceptualizację projektową. Została wyspecyfikowana również syntaktyka języka projektowania i realizacja tej syntaktyki wraz z dziedziną diagramów projektowych umożliwiająca otrzymywanie diagramów projektowych. Proponowany system wspomagający projektowanie wykorzystujący opisany tu model formalny języka projektowania ułatwia projektantowi interakcję z diagramami, pomaga mu uniknąć błędów, jak również w możliwie krótkim czasie osiągnąć wyznaczone cele projektowe.

Rozdział trzeci opisuje wewnętrzną reprezentację diagramów w postaci atrybutowanych hierarchicznych hipergrafów, gdzie atrybuty przypisane atomom hipergrafów przechowują semantyczną wiedzę projektową. Wykorzystywany tu formalny model hierarchicznych hipergrafów został wprowadzony w [DrHP00] i rozszerzony w [Ślus03] w celu wyrażania relacji wieloargumentowych pomiędzy elementami znajdującymi się na różnych poziomach hierar-

chii. Stosowane tu hipergrafy mogą być także traktowane jako rozszerzenie hipergrafów używanych w [Mina02], które są zbyt restrykcyjne w sposobie wyrażania relacji projektowych.

Hierarchiczne hipergrafy odzwierciedlają sposób projektowania "od ogółu do szczegółu" i pozawalają projektantowi rozważać projekt na wybranym poziomie szczegółowości. Hiperkrawędzie, z których składają się hipergrafy, reprezentują zarówno komponenty diagramów, jak i wieloargumentowe relacje między nimi. Hierarchiczne hiperkrawędzie reprezentują grupy komponentów diagramu, natomiast zagnieżdżone w nich hipergrafy odpowiadają częściom składowym komponentów diagramu. Taka reprezentacja wewnętrzna diagramów daje możliwość przechowywania zarówno syntaktycznej, jak i semantycznej wiedzy projektowej i dzięki temu pozwala systemowi wnioskować o projektach i wspomagać użytkownika poprzez sugerowanie mu kolejnych kroków projektowych oraz ostrzeganie przed tworzeniem rozwiązań niezgodnych z określonymi wymaganiami lub kryteriami projektowymi.

Rozdział czwarty przedstawia operacje zdefiniowane na hierarchicznych hipergrafach, które odpowiadają poszczególnym akcjom projektowym wykonywanym przez użytkownika na diagramie. Pierwszy narysowany przez użytkownika diagram projektowy jest automatycznie przekształcany na reprezentację hipergrafową. Wszystkim modyfikacjom diagramu będącym odzwierciedleniem nowych idei projektowych [DoGr01] i dokonywanym przez użytkownika w trakcie procesu projektowania odpowiadają zmiany w hipergrafowej strukturze danych definiowane za pomocą operacji na hipergrafach [Ślus03].

Podstawową operacją wykonywaną na hipergrafie podczas procesu projektowego jest *rozwinięcie hiperkrawędzi* zdefiniowane w [GLŁŚ07]. Operacja ta odzwierciedla podział wybranego fragmentu na mniejsze części, pozwalając reprezentować strukturę projektowanego obiektu w sposób bardziej szczegółowy. Zagnieżdża ona hierarchiczny hipergraf reprezentujący składowe części komponentu i relacje między nimi w hiperkrawędzi danego hipergrafu reprezentującej uszczegółowiany komponent. Operacją odwrotną do rozwijania hiperkrawędzi jest operacja usunięcia jej zawartości zdefiniowana w [GrŚL08]. Jest ona szczególnie użyteczna w przypadku projektowania złożonych obiektów, ponieważ pozwala usuwać podziały wybranych części i rozważać wybrane komponenty obiektu na wyższym poziomie szczegółowości. Operacja ta usuwa hipergraf zagnieżdżony wcześniej w jakiejś hiperkrawędzi reprezentującej składowy komponent projektowanego obiektu.

W niniejszej pracy zostaną zdefiniowane kolejne dwie wzajemnie odwrotne operacje wykonywane na hierarchicznych hipergrafach. Będzie to operacja konkatenacji, która pozwoli dodać do istniejącego hierarchicznego hipergrafu nowy hipergraf reprezentujący strukturę nowego fragmentu projektowanego obiektu, oraz operacja usunięcia podgrafu hierarchicznego hipergrafu odpowiadająca likwidacji wybranej części modelowanego artefaktu. Operacje wykonywane na wewnętrznej reprezentacji wizualnego języka projektowania nie tylko po-

zwalają systemowi projektowemu tworzyć i modyfikować hipergrafy reprezentujące diagramy, ale także stanowią narzędzie wspomagające dynamiczną wizualizację tych diagramów.

Rozdział piąty omawia hierarchiczne gramatyki hipergrafowe, będące efektywnym narzędziem służącym do generacji hierarchicznych hipergrafów [Ślus99, Ślus04]. Produkcje gramatyki stosowane w kolejnych krokach wywodu pozwalają otrzymać hipergraf reprezentujący strukturę rozwiązania danego problemu projektowego. Zostały zaprezentowane podstawy teoretyczne dotyczące hierarchicznych gramatyk rozmieszczenia, bezkontekstowych hierarchicznych gramatyk rozmieszczenia i programowanych hierarchicznych gramatyk rozmieszczenia. Wykorzystywane tu programowane gramatyki hipergrafowe wyposażone są w diagram sterujący, który kontroluje sposób przeszukiwania przestrzeni dopuszczalnych rozwiązań danego problemu projektowego.

W rozdziale szóstym został opisany logiczny model wnioskowania o projektach z wykorzystaniem wewnętrznej reprezentacji diagramów projektowych. Diagramy są tu opisywane za pomocą formuł logicznych pierwszego rzędu, których syntaktyka bazuje na wielorodzajowej sygnaturze języka logicznego. Ontologiczne dopasowanie symboli sygnatury do pojęć konceptualizacji projektowej umożliwia jednoznaczne przedstawienie wiedzy dotyczącej obiektów projektowych reprezentowanych przez komponenty diagramów. Semantyka formuł logicznych jest określana za pomocą interpretacji definiowanej poprzez strukturę relacyjną przypisującą atomy hipergrafów i wartości ich atrybutów elementom składowym formuł. Interpretacja ta umożliwia pozyskiwanie wiedzy projektowej, dotyczącej diagramów i śledzenie zmian zachodzących w projektach w wyniku wykonywania kolejnych akcji projektowych.

Utworzony zbiór formuł logicznych specyfikujących własności diagramów jest porównywany ze zbiorem formuł logicznych pierwszego rzędu opisujących kryteria projektowe. Ograniczeniom i wymaganiom projektowym wyrażanym jako formuły logiczne odpowiadają określone warunki, które powinny być spełnione przez wewnętrzną reprezentację diagramów. Porównywanie formuł logicznych reprezentujących ograniczenia projektowe z formułami opisującymi diagramy bądź sprawdzanie spełnialności wyspecyfikowanych warunków przez hierarchiczne hipergrafy odpowiada stosowaniu dobrze określonych zasad projektowania inżynierskiego lub architektonicznego w celu ułatwienia projektantowi efektywnej realizacji zakładanych celów projektowych. Tak więc wiedza projektowa przechowywana w hipergrafowej reprezentacji języka wizualnego pozwala systemowi projektowemu wspomagać koncepcyjną fazę projektowania poprzez wnioskowanie dotyczące poszczególnych elementów języka, reprezentujących zarówno rozwiązania kompletne, jak i rozwiązania częściowe.

Rozdział siódmy przedstawia system inteligentnych agentów wspomagających projektowanie z wykorzystaniem wiedzy projektowej. Inteligentni agenci potrafią generować i modyfikować reprezentację środowiska na podstawie doświadczeń płynących z interakcji z pro-

jektantem i innymi agentami, sugerować projektantowi oryginalne rozwiązania, jak również oceniać rozwiązania częściowe wygenerowane w trakcie procesu projektowania [Wood95]. W ostatnich latach znaczna część badań naukowych dotyczących projektowania jest skoncentrowana na agentowych systemach projektowych [MyPo94, Land97, Cana97, Saun01, GrŚG05, Dzer02]. Adaptacyjne metody agentowe wykorzystywane w projektowaniu konceptualnym są przedstawione w [CaCK98], podczas gdy systemy projektowe z uczącymi się agentami w [GrBr00]. Systemy agentowe są też łączone z różnymi typami gramatyk [PaSa99, McCa02]. Systemy te nie są jednak na tyle elastyczne, aby w zadowalający sposób poradzić sobie z dynamiczną naturą projektowania, co jest szczególnie ważne w jego koncepcyjnej fazie.

Zaproponowany wieloagentowy system projektowy jest modularnym systemem współbieżnym wspomagającym projektanta w rozwiązywaniu skomplikowanych zadań projektowych. Architektura tego systemu składa się ze zbioru komunikujących się ze sobą agentów, którzy autonomicznie rozwiązują powierzone sobie podzadania, operując na hierarchicznej hipergrafowej reprezentacji projektów. Zachowanie agentów jest modelowane za pomocą zbioru reguł hierarchicznej gramatyki hipergrafowej. Każdy agent posiada zarówno odpowiednią gramatykę hipergrafową, jak i bazę wiedzy zawierającą fakty dotyczące podzadania, za które jest on odpowiedzialny. Na przebieg procesu projektowego ma tu wpływ interakcja pomiędzy wyspecjalizowanymi, współpracującymi ze sobą agentami. Dynamicznie zmieniający się kontekst projektowy wyraża zarówno środowisko, w którym działają agenci, jak i predykaty opisujące kryteria projektowe. Jest tu przedstawiony zarówno semantyczny model wieloagentowego systemu projektowego wykorzystującego hierarchiczne gramatyki rozmieszczenia, jak też pojęcia poprawnego rozwiązania projektowego i rozwiązania zgodnego z kryteriami projektowymi. Do sprawdzania zgodności rozwiązań z kryteriami projektowymi jest wykorzystywany mechanizm wnioskowania bazujący na modalnym rachunku predykatów.

Dzięki wykorzystywanej reprezentacji projektowanych obiektów w postaci hierarchicznych hipergrafów agenci mogą się w łatwy sposób adaptować do zmian specyfikacji problemu projektowego zachodzących w trakcie projektowania i realizować adaptacyjne poszukiwanie rozwiązań. Wiedza projektowa zakodowana w atrybutowanych hierarchicznych hipergrafach pozwala im zachowywać się w sposób dynamiczny dzięki możliwości dokonywania oceny rozwiązań częściowych reprezentujących aktualny stan projektowanego obiektu i podejmowania na tej podstawie kolejnych decyzji projektowych. Agenci planują swoje przyszłe zachowanie poprzez wybór produkcji gramatyki, które należy zastosować w kolejnych krokach procesu.

W rozdziale ósmym są omówione dwa systemy komputerowe wspomagające projektowanie koncepcyjne, które zostały zaimplementowane w Zakładzie Projektowania i Grafiki

Komputerowej UJ w sposób zgodny z założeniami przedstawionymi w zaprezentowanym w pracy modelu teoretycznym. Pierwszy z nich to prototypowy system wspomagania projektowania i wnioskowania *HSSDR* (Hypergraph System Supporting Design and Reasoning) zaimplementowany w Javie przez Szymona Gajka [GBPG09, Gaje11]. Wykorzystuje on wiedzę projektową do wspomagania użytkownika, który projektuje obiekty z użyciem diagramów. Stosowane tu diagramy projektowe tworzą diagramowy język wizualny. System ten pośredniczy pomiędzy językiem diagramowym a jego wewnętrzną reprezentacją w postaci hierarchicznych hipergrafów, daje możliwość definiowania ograniczeń projektowych i wnioskowania dotyczącego diagramów.

System *HSSDR* składa się z pięciu modułów: *interfejsu graficznego* pozwalającego tworzyć i edytować diagramy, ich atrybuty i związane z nimi ograniczenia, *generatora hierarchicznych hipergrafów* tworzącego hipergrafy i stosującego do nich operacje odpowiadające akcjom projektowym wykonywanym przez użytkownika na diagramach, *modułu wnioskowania* pozwalającego wnioskować o diagramach poprzez sprawdzanie zgodności opisujących je formuł z formułami logicznymi wyrażającymi ograniczenia projektowe, *modułu sterującego wizualizacją hipergrafów*, który uaktualnia i weryfikuje istniejącą hierarchię, pozwala grupować komponenty i umożliwia nawigację pomiędzy różnymi poziomami hierarchii hipergrafu, oraz *kontrolera* zapewniającego synchronizację pomiędzy interfejsem graficznym a reprezentacją wewnętrzną.

Drugi z omawianych programów nosi nazwę *Arrange Editor* i jest prototypowym wieloagentowym systemem projektowym zaimplementowanym w języku Java przez Tomasza Zarasia [Zara09]. System ten składa się z dwóch modułów. Pierwszy z nich jest *edytorem* pozwalającym projektantowi definiować hierarchiczne gramatyki rozmieszczenia modelujące zachowanie agentów i dodawać do bazy wiedzy ograniczenia dotyczące prymitywów geometrycznych reprezentujących komponenty obiektu. Drugi moduł jest *generatorem* tworzącym wizualizacje rozwiązań odpowiadających hipergrafom wygenerowanym przez cały system lub poszczególnych agentów. Po uruchomieniu systemu projektant może śledzić pracę różnych agentów przechodząc pomiędzy przypisanymi im oknami, w których są widoczne generowane przez nich hipergrafy i ich graficzne modele. Wynik pracy całego systemu jest prezentowany w oknie przypisanym agentowi-menadżerowi.

Wykorzystanie zaproponowanego formalnego modelu komputerowego wspomagania projektowania wizualnego z wykorzystaniem wiedzy projektowej jest tu przedstawione na przykładach projektowania rozkładów pomieszczeń, rozmieszczania mebli i sprzętów w pomieszczeniach oraz projektowania krzeseł i kratowych wież przesyłowych. Przykłady dotyczące projektowania rozkładów pomieszczeń są ilustrowane rysunkami uzyskanymi za pomocą systemu *HSSDR*, a rysunki wykorzystywane w przykładach dotyczących aranżacji pomieszczeń pochodzą z programu *Arrange Editor*. Rysunki ilustrujące przykłady dotyczące

projektowania krzeseł oraz wież przesyłowych zostały wykonane w programach napisanych również w Zakładzie Projektowania i Grafiki Komputerowej UJ przez Piotra Nikodema [Niko01, NiSt04]. Programy te są zaimplementowane w Javie i pozwalają projektować wspomniane artefakty z wykorzystaniem ich hierarchicznej grafowej reprezentacji wewnętrznej.

Przedstawione w rozprawie narzędzia zastosowane w prototypowych systemach wspomagania projektowania opartych na wiedzy projektowej wskazują na możliwość efektywnego asystowania w procesie innowacyjnego projektowania poprzez ułatwienie projektantowi szybkiej generacji i modyfikacji wstępnych rozwiązań projektowych oraz wnioskowanie na temat ich poprawności. Systemy wspomagające koncepcyjną fazę procesu projektowego znajdą zastosowanie w projektowaniu architektonicznym i aranżacji wnętrz, a także w projektowaniu przedmiotów użytkowych i urządzeń mechanicznych.

2. DIAGRAMOWY JĘZYK WIZUALNY

W tym rozdziale zostanie omówiona ogólna charakterystyka języków wizualnych, a także ich rola w projektowaniu wizualnym. Następnie zostanie zdefiniowana zaproponowana w pracy konceptualizacja dziedziny projektowania wizualnego, będąca specyfikacją pojęć z tej dziedziny, ich taksonomii oraz zachodzących między nimi relacji. Konceptualizacja ta umożliwia ontologiczne ujęcie modelu komputerowego wspomagania projektowania. Zostaną także przedstawione reguły przekształcania struktur projektów złożonych z obiektów określonych poprzez konceptualizację projektową. Zostanie również wprowadzona syntaktyka języka projektowania i jej realizacja umożliwiająca otrzymywanie diagramów projektowych będących elementami wizualnego języka projektowania.

2.1. Języki wizualne

Język wizualny zawiera zbiór zasad, dzięki którym obrazy służą przekazywaniu koncepcji i wymianie informacji. Jest on odpowiednikiem języka naturalnego, w którym zamiast słów liniowej postaci są stosowane elementy obrazu reprezentujące idee w kontekście przestrzennym. Każdy człowiek posiada wrodzoną zdolność kognitywnego modelowania idei i wyrażania ich poprzez szkice, rysunki, konstrukcje i akcje. Według L.B. Archera [Arch79] sposób, w jaki ludzie tworzą w swoim umyśle obrazy manipulując ideami i oceniając je przed, podczas i po ich uzewnętrznieniu określa system poznawczy człowieka.

Dzięki systemowi poznawczemu jest możliwa komunikacja wizualna za pomocą wyrażeń wizualnych odnoszących się do różnych dziedzin życia. Zarówno w komunikacji wizualnej, jak i wnioskowaniu wizualnym niezwykle istotny jest odpowiedni dobór graficznej reprezentacji danych i określenie znaczącej interpretacji pozwalającej odzwierciedlać intencje kryjące się za odpowiednim wykorzystaniem obrazów. Użycie właściwej reprezentacji obiektów i relacji zachodzących między nimi często ułatwia przedstawienie pewnych cech rozważanego problemu, gdyż pozwala na jawne wyrażenie i bezpośredni dostęp do takich elementów danych, które mogą być reprezentowane tylko niejawnie w innych typach reprezentacji.

Jak zauważył Wittgenstein [Witt88], znaczenie wyrażeń wizualnych jest zdeterminowane poprzez sposób ich użycia. Dlatego też interpretacja określająca znaczenie stosowanych wyrażeń wizualnych i umożliwiająca wnioskowanie wizualne nie powinna być traktowana jako część specyfikacji obrazu, ale jako niezależne odwzorowanie przypisujące wyrażeniom wizualnym znaczenie odzwierciedlające różne sposoby ich użycia [WaLe93].

Obrazy pomagają nam komunikować się i wnioskować, gdyż interpretacja może im nadać znaczenie w terminach problemów dotyczących określonej dziedziny zastosowań. Wnioski dotyczące przestrzennych własności obrazów są wówczas przenoszone na problemy z danej dziedziny. Tak więc wnioskowanie z obrazów, czyli tzw. wnioskowanie graficzne, pozwala nam wnioskować o problemach reprezentowanych przez wyrażenia wizualne. Przykładem mogą być tutaj diagramy Venny wykorzystywane do wnioskowania, dotyczącego praw algebraicznych w teorii zbiorów.

Języki wizualne zyskują obecnie coraz większe znaczenie w interakcji między człowiekiem i komputerem [RMRF96]. Są więc tworzone formalne modele specyfikacji języków wizualnych, opisu obrazów w tych językach, specyfikacji interpretacji obrazów i wnioskowania wspomaganego przez zinterpretowane obrazy. Techniki definiowania języków wizualnych różnią się sposobem, w jaki opisują syntaktyczne i semantyczne aspekty tych języków oraz wykorzystywaną reprezentacją wizualną. W podejściu lingwistycznym syntaktyczne własności języków wizualnych są opisywane z wykorzystaniem produkcji gramatyk grafowych [ReSc97, Schu91] lub hipergrafowych [Mina97]. Metody te pozwalają na bezpośrednią manipulację konceptualnymi konstrukcjami języka [Mina02, ScWZ95]. Gramatyki grafowe są także wykorzystywane do definiowania semantyki języków wizualnych [BaEh99, ErRT99]. Drugie podejście do specyfikacji języków wizualnych oparte jest na formalizmach logicznych opisujących możliwe topologiczne relacje między obiektami [GoCo96]. Teoria opisu i wnioskowania z notacji wizualnych, która bazuje na logice, jest przedstawiona w [Haar99]. Natomiast algebraiczne podejście do specyfikacji języków wizualnych polega na zdefiniowaniu funkcji, które tworzą złożone obrazy z ich podstawowych elementów. Analiza relacji pomiędzy wizualną reprezentacją i strukturą przez nią reprezentowaną oparta na podejściu algebraicznym jest przedstawiona w [Gurr98]. Sterowane składnią algebraiczne podejście do semantyki obrazu jest opisane w [WaZe98]. Przegląd specyfikacji języków wizualnych znajduje się w [MeMW98].

Diagramy wizualne są wykorzystywane w wizualnym procesie projektowym opisanym w [ChHY97], a także w systemie TAC (The Architect's Collaborator) [Koil04] będącym prototypowym narzędziem projektanta wspomagającym iteracyjne modyfikowanie projektów. Język wizualny, zwany LSD [CoSm98], jest wykorzystywany do projektowania rodziny obiektów o określonej strukturze. W [CoSm00] język ten został uogólniony do opisu sparametryzowanych brył i wykonywanych na nich operacji. W [BaCo05] jest przedstawione wy-

korzystanie języka LSD do sterowania syntezą projektów będących konfiguracjami sparametryzowanych obiektów strukturalnych. Język wizualny umożliwiający specyfikację reguł gramatyki kształtu wykorzystywanych do generacji rozkładów pomieszczeń zgodnych z wymaganiami użytkownika jest opisany w [RCSm97]. Język ten stanowi podstawę systemu automatycznej generacji projektów domów [RMDG97]. To narzędzie projektowe umożliwia użytkownikowi wybranie symbolicznego opisu domu wygenerowanego za pomocą gramatyki kształtu, dostosowanie go do własnych potrzeb z zastosowaniem edytora graficznego i stworzenie modelu zaprojektowanego domu. Wymienione tu systemy projektowe są jednakże albo systemami eksperymentalnymi operującymi na prostych przykładach, albo nie posiadają odpowiedniej reprezentacji wiedzy projektowej, która umożliwiałaby wspomaganie wnioskowania o generowanych projektach.

W [AkMo04, Mous04] została wprowadzona notacja ICE (Interactive Configuration Exploration) pozwalająca w sposób formalny opisać generację i modyfikowanie kształtów i form występujących w rysunkach projektowych. Podstawowymi jednostkami syntaktycznymi notacji ICE są *regulatory*, które składane i łączone ze sobą na różne sposoby pozwalają kontrolować zmiany zachodzące w rozwiązaniu projektowym. Regulatory transformacji są wykorzystywane do generacji elementów wizualnych i ich złożonych konfiguracji. Pozostałe regulatory (ograniczeń, hierarchii, operacji i wariacji) stanowią tzw. regulatory relacyjne sterujące konfiguracją, parametrami lub atrybutami elementów powstałych z wykorzystaniem regulatorów transformacji. Notacja ICE pozwala wyrazić zarówno historię powstawania projektu, jak też dostarcza informacji o zbiorze zastosowanych przekształceń. Nie umożliwia ona jednak wyrażania zmian relacji z elementami zewnętrznymi względem elementów modyfikowanych przez regulatory. System ICE, oferując funkcjonalności niezależne od konkretnych problemów projektowych, nie operuje pojęciami specyficznymi dla danej dziedziny projektowania.

Obecny rozwój badań dotyczących języków wizualnych i wnioskowania wizualnego zmierza w kierunku tworzenia systemów komputerowego wspomagania wykorzystania obrazów jako znaczących wyrażeń wizualnych w różnych dziedzinach zastosowań, jak np. projektowanie wizualne [Grab07], wizualne bazodanowe języki zapytań [Stas08] czy programowanie wizualne [AjTs08]. Jednym z istotnych problemów jest odpowiednie wyposażenie tych systemów w interpretację. Interpretacja wyrażeń wizualnych, takich jak ikony czy diagramy może być a priori zdefiniowana przez projektanta języka lub specyfikowana bezpośrednio przez użytkownika systemu.

W pierwszym przypadku zakłada się, że wbudowane znaczenie wyrażeń wizualnych jest rozumiane w sposób uniwersalny i jest odpowiednie do zastosowań w dziedzinach, gdzie powszechnie stosuje się metody graficznej reprezentacji. Przykład stanowi wykorzystanie wyrażeń wizualnych, takich jak diagramy przepływu danych do reprezentacji procesów obli-

czeniowych w wizualnych językach programowania [Hils92] lub ikon do reprezentacji klas obiektów w bazach danych [TYHT90]. W drugim przypadku użytkownik może sam określać pojęcia za pomocą obrazów i definiować klasy obrazów dla wybranej reprezentacji w taki sposób, że odpowiednie znaczenie jest związane ze specyfikacją tych obrazów.

Wybór reprezentacji wizualnej w systemie wspomagającym wykorzystanie obrazów w dużej mierze zależy od aktualnie rozważanego problemu. Ponadto, w systemie takim nie tylko musi być wyspecyfikowana klasa obrazów, ale należy także nadać znaczenie operacjom wykonywanym na obiektach graficznych. Bowiem we wnioskowaniu bazującym na wyrażeniach wizualnych informację niosą nie tylko całe obrazy, ale także ich graficzne składowe (podobrazy) i obrazy emergentne oraz ich własności. Wszystkie one powinny mieć nadane odpowiednie znaczenie, co pozwoli na efektywne prowadzenie wnioskowania wspomaganego poprzez interpretację obrazów.

2.2. Języki wizualne w projektowaniu

Inspiracją do powstania lingwistycznych metod projektowania była analogia pomiędzy projektowaniem a językami naturalnymi. W metodach tych podstawowe terminy dotyczące języków naturalnych przenosi się do projektowania. Składowe projektu będące elementami słownika odpowiadają słowom z języka. Konfiguracje składowych projektu mogą być traktowane jak zdania budowane ze słów. Na podstawie słownika elementów projektu i wiedzy projektowej można utworzyć narzędzie generacyjne odpowiadające gramatyce języka. Analogią do interpretacji projektu jest semantyka języka [CRRB90].

Definiując model projektowy wykorzystywany w projektowaniu wspomaganym komputerowo, wyróżnia się:

- *słownik* zawierający elementy zastosowane w projekcie,
- *interpretację*, która może być
  - *zamierzona* – tzn. określająca cele i wymagania używane do sterowania procesem projektowym, lub
  - *faktyczna* – tzn. określająca ocenę gotowego projektu,
- *wiedzę projektową*, będącą zasobem wiadomości z projektowania wykorzystywanym przez projektanta, lub system projektowy (zawiera ona wiedzę o interpretacji i wiedzę o składni),
- *opis projektu* stanowiący rezultat procesu projektowania, będący wynikiem decyzji projektowych.

Proces projektowy jest scharakteryzowany za pomocą *generacji* i *interpretacji*. Generowanie projektów odbywa się na podstawie słownika oraz wiedzy syntaktycznej związanej

z rozmieszczeniem elementów ze słownika. W wyniku generacji otrzymuje się opisy projektów. Interpretacja natomiast jest funkcją opisu projektu.

Rozważając projektowanie diagramowe definiuje się słownik w postaci zbioru kształtów elementarnych, z których poprzez transformacje otrzymuje się składowe diagramów projektowych. Diagramy projektowe stanowią klasę obiektów wizualnych, będących konfiguracjami skończonej liczby kształtów elementarnych i stanowiących opisy projektów. W projektowaniu diagramowym proces projektowy jest związany z procesem generacji i modyfikacji diagramów poprzez dodawanie do niego nowych i zmianę istniejących elementów. W konkretnych zastosowaniach nie używa się uniwersalnego słownika prymitywów graficznych, ale w definicji takiego słownika uwzględnia się dziedzinę zastosowań. Wiedza projektowa zorientowana problemowo ma więc wpływ na rodzaj wykorzystywanych diagramów. Oznacza to, że zadanie projektowe determinuje używany język wizualny.

Wprowadzenie syntaktycznie i semantycznie poprawnego diagramowego języka projektowania umożliwia zastosowanie metod syntaktycznych w projektowaniu. Elementami słownika są tu kształty odpowiadające wizualnym komponentom diagramów, natomiast syntaktyczne reguły projektowe pozwalają odpowiednio przekształcać diagramy. Semantyczna wiedza przechowywana pod postacią atrybutów związanych z komponentami diagramów umożliwia niezwykle istotne dla projektanta automatyczne wnioskowanie o wymaganych własnościach projektów.

2.3. Diagramowy język projektowania

Diagramowy Język Projektowania (*DJP*) powinien pozwalać projektantowi wyrazić jak największą liczbę potencjalnych rozwiązań. Dlatego język taki jest tworzony zgodnie z zasadami umożliwiającymi zawarcie w nim elementów, z których każdy reprezentuje całą klasę rozwiązań projektowych, spełniających określone wymagania. Diagramowy język projektowania jest więc zbiorem uogólnionych diagramów projektowych, które zawierają wyabstrahowaną informację o projektowanym obiekcie, czyli taką, która jest zredukowana do podstawowych składowych wizualnych wyrażających najbardziej istotne cechy niezbędne do opisu znaczenia [Grab07].

Jednoznaczne wykorzystanie języka diagramowego jest możliwe dzięki podejściu ontologicznemu, którego elementem jest zdefiniowanie konceptualizacji dziedziny projektowej. Konceptualizacja ta pozwala określić zbiór obiektów, które mogą stanowić elementy rozwiązania zadania projektowego, ich hierarchię oraz zbiór relacji, jakie mogą zachodzić pomiędzy obiektami. Następnie na podstawie wymagań projektowych są ustalane struktury projektów, stanowiące syntaktykę diagramowego języka projektowania. Umożliwia ona tworzenie dia-

gramów i stanowi podstawę do nadawania im znaczenia, ich interpretacji i wnioskowania diagramowego.

Ponieważ elementy *DJP* dla każdej dziedziny zastosowań są definiowane w inny sposób, język ten może być traktowany jako rodzina języków projektowych. W niniejszej pracy diagramowy język projektowania będzie wykorzystywany do projektowania rozkładów pomieszczeń, krzeseł oraz topologii kratowych wież przesyłowych. Po przypisaniu elementom konceptualizacji projektowej obiektów i relacji specyficznych dla każdej z tych dziedzin dostaniemy konceptualizacje zadaniowe, a oparte na nich języki staną się zorientowane problemowo. Otrzymamy odpowiednio diagramowy język rozmieszczenia pomieszczeń, diagramowy język opisujący krzesła i diagramowy język wież przesyłowych. Języki te służą projektantowi jako narzędzia do komunikacji z systemem projektowym wykorzystującym wiedzę z danej dziedziny i pozwalają mu wykonywać określone akcje na diagramach.

Wprowadzone w pracy pojęcia będą ilustrowane na trzech przykładach, z których pierwszy dotyczy projektowania rozkładów pomieszczeń, drugi – projektowania krzeseł, a trzeci – projektowania kratowych wież przesyłowych. Wieża kratowa jest konstrukcją szkieletową, której optymalizacja może być rozważana na trzech poziomach: topologii, geometrii i poszczególnych komponentów [BGNS03]. Wybór podstawowych elementów i sposób ich połączenia jest rozważany na poziomie topologicznym. Na poziomie geometrii komponenty struktury są rozmieszczane w przestrzeni w optymalny sposób. Na ostatnim poziomie poszukuje się najlepszych atrybutów dla poszczególnych elementów struktury, jak przekroje prętów skratowania czy parametry wykorzystywanego materiału, m. in. na podstawie numerycznych symulacji rozkładu naprężeń w projektowanej konstrukcji.

Problem optymalizacji topologicznej wież przesyłowych był rozważany w [NiSt04]. Struktury wież były reprezentowane za pomocą hierarchicznych grafów kompozycyjnych [BGNS03], a do optymalizacji ich topologii zostały zastosowane metody ewolucyjne. W pierwszym kroku była optymalizowana sama struktura wieży, a następnie optymalizowano wybraną strukturę pod względem liczby i układu elementów skratowania. Początkowa populacja struktur optymalizowanych w pierwszym etapie była generowana za pomocą gramatyki grafowej.

Wieże przesyłowe zaprojektowane z wykorzystaniem omawianego tu języka diagramowego mogą stanowić populację początkową na obu etapach optymalizacji topologicznej projektowanej konstrukcji stworzoną z uwzględnieniem zadanych ograniczeń (np. szerokość wieży) bądź też być podstawą do stworzenia gramatyki grafowej automatycznie generującej populację początkową.

2.3.1. Diagramy projektowe

Diagramy stanowią najczęściej stosowaną w projektowaniu kategorię obiektów wizualnych. Służą one projektantowi do strukturalizacji i wizualizacji idei projektowych. Diagramy są konfiguracjami skończonej liczby składowych będących elementarnymi kształtami rozmieszczonymi przestrzennie za pomocą określonego zbioru dopuszczalnych transformacji. Każda dziedzina projektowania specyfikuje konwencje związane z wykorzystywanymi składowymi diagramów i determinuje w ten sposób stosowany język wizualny. Diagramy projektowe są to więc takie diagramy wizualne, które opisują wykreowane rozwiązania zadań projektowych. Diagramowy język projektowania może być zdefiniowany jako wybrany podzbiór diagramów wizualnych.

Definicja 2.1:
Niech S będzie zbiorem ograniczonych podzbiorów $\mathfrak{R}^n$, zwanych kształtami elementarnymi. Niech F będzie zbiorem dopuszczalnych transformacji z $\mathfrak{R}^n$ w $\mathfrak{R}^n$.

Diagramowy Język Projektowania (*DJP*) jest podzbiorem diagramów, z których każdy jest sumą skończonej ilości elementów z S przetransformowanych z wykorzystaniem odwzorowań z F, tzn. $DJP \subseteq \{\bigcup_{i=1}^{n} f_i(s_i) \mid s_i \in S, f_i \in F\}$. □

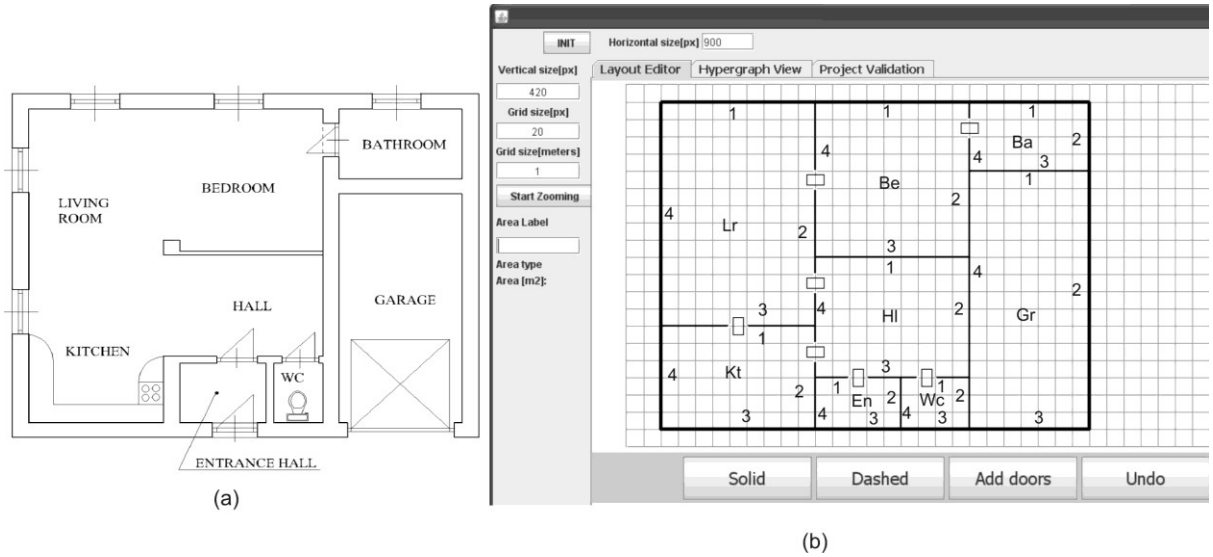
W procesie wizualnego projektowania diagramy projektowe są przekształcane w sposób umożliwiający reprezentowanie projektowanego obiektu na różnych poziomach szczegółowości. Diagramowy język projektowania odgrywa podobną rolę w projektowaniu co szkice projektowe, gdyż umożliwia projektantowi śledzenie zmian dokonywanych w diagramach projektowych. Wykorzystanie diagramów projektowych jako formalnego języka reprezentacji wiedzy pozwala zdefiniować komputerowo wspomagane metody syntezy rozwiązań projektowych oraz wnioskowania o nich opartego na analizie diagramów.

Przykład 2.1: rozkłady pomieszczeń
Rozważmy diagramowy język służący do projektowania rozkładów pomieszczeń, którego prototypowa wersja została opracowana w grancie [Spra06]. Elementy tego języka mają postać diagramów, które mogą być postrzegane jako uproszczone rysunki architektoniczne.

Wykorzystywane tu diagramy projektowe reprezentujące rozkłady pomieszczeń składają się z kształtów elementarnych będących wielokątami. Wielokąty te za pomocą odpowiednich transformacji są wpisywane w ortogonalną siatkę. Pozwala to odnieść ściany wielokątów do czterech kierunków głównych (północ, południe, wschód, zachód). Wielokąty te reprezentują komponenty rozkładu pomieszczeń, takie jak obszary użytkowe i pokoje. Boki każdego z wielokątów są uporządkowane zgodnie z ruchem wskazówek zegara, zaczynając od najbardziej lewego górnego. Wzajemna lokalizacja wielokątów jest wyznaczona na podstawie kry-

teriów projektowych. Relacja dostępności pomiędzy komponentami jest reprezentowana za pomocą linii z małymi prostokątami. Linia taka może odpowiadać ścianie z drzwiami, fragmentowi ściany lub ścianie nieistniejącej fizycznie. Relacja przyległości między komponentami jest oznaczana liniami ciągłymi wspólnymi dla sąsiednich wielokątów.

Przykładowy rozkład pomieszczeń i reprezentujący go diagram projektowy są przedstawione na rys. 2.1. Rozkład pomieszczeń zawiera osiem prostokątów reprezentujących salon (*Lr*), kuchnię (*Kt*), sypialnię (*Be*), łazienkę (*Ba*), hol (*Hl*), wejście (*En*), toaletę (*Wc*) i garaż (*Gr*). Pomiędzy tymi komponentami są określone dwa typy relacji: dostępność i przyległość. Przykładowo, istnieje przejście pomiędzy kuchnią i salonem, a nie ma bezpośredniego przejścia między holem i sypialnią, które ze sobą sąsiadują. ◊



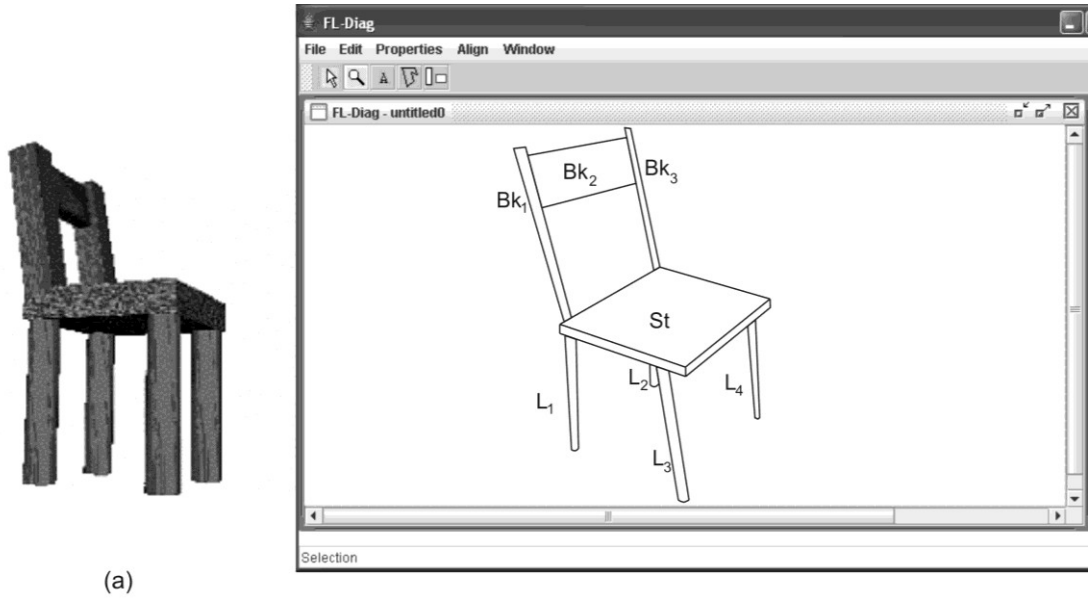
Rys. 2.1. Rozkład pomieszczeń: a) rysunek architektoniczny, b) diagram projektowy w programie *HSSDR*

Fig. 2.1. A floor layout: a) an architectural drawing, b) a design diagram in *HSSDR* program

Diagramowy język projektowania może być także wykorzystany na przykład w dziedzinie wzornictwa przedmiotów użytkowych lub projektowania konstrukcji stalowych. Rozważymy tutaj przykłady projektowania krzeseł i kratowych wież przesyłowych.

Przykład 2.2: krzesła

Diagramy projektowe reprezentujące modelowane krzesła składają się z dwuwymiarowych rzutów odpowiednio przetransformowanych podstawowych brył, takich jak prostopadłościany, walce, stożki i kule. Bryły te reprezentują komponenty krzeseł, a więc siedziska, elementy oparcia i elementy postawy. Pomiędzy poszczególnymi komponentami bądź ich fragmentami występuje jedynie relacja przyległości. Model krzesła i diagram służący do jego zaprojektowania są przedstawione na rys. 2.2. Krzesło to składa się z czterech prostopadłościanów (siedzisko (*St*) i trzy części oparcia (Bk_1-Bk_3)) oraz czterech walców (nogi (L_1-L_4)). ◊



Rys. 2.2. Krzesło: a) trójwymiarowy model, b) diagram projektowy
Fig. 2.2. A chair: a) a three-dimensional model, b) a design diagram

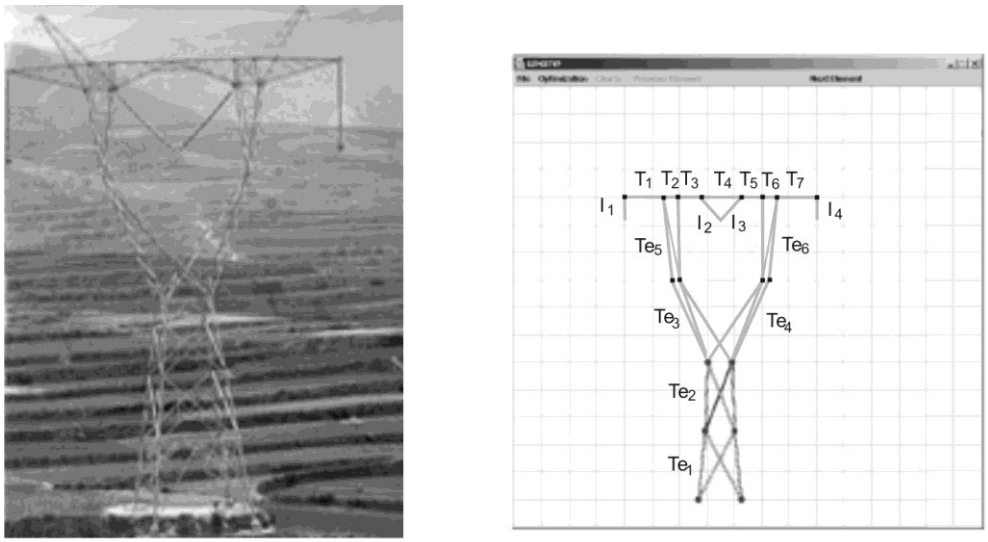
Przykład 2.3: wieże przesyłowe

Diagramy projektowe reprezentujące wieże przesyłowe składają się z przetransformowanych dwuwymiarowych elementów odpowiadających fragmentom skratowania trzonu i gałęzi wieży, odcinków poziomych reprezentujących skratowania poziome i odcinków pionowych lub ukośnych odpowiadających izolatorom. Pomiędzy węzłami elementów skratowania zachodzi jedna relacja reprezentująca zespolenie tych elementów oraz ich połączenie z izolatorami. Wieża przesyłowa i diagram projektowy reprezentujący jej widok z przodu są przedstawione na rys. 2.3. Dolny segment wieży przedstawionej na diagramie tworzy skratowanie trzonu złożone z dwóch elementów (Te_1, Te_2), środkowy segment składa się z dwóch dwuelementowych skratowań gałęzi wieży (Te_3-Te_6), a segment górny zawiera siedem skratowań (T_1-T_7) i cztery izolatory pozwalające na podwieszenie kabli (I_1-I_4). ◊

2.3.2. Syntaktyka diagramowego języka projektowania

Dla określonego typu zadań projektowych jest definiowana syntaktyka diagramowego języka projektowania zorientowanego na ten typ zadań. Stanowi ona podstawę do tworzenia diagramów projektowych, nadawania im znaczenia, interpretowania ich oraz wnioskowania z nich. Początkowo jest określana konceptualizacja [GuOS09], będąca formalną reprezentacją wiedzy projektowej [CRRB90] dotyczącej rozwiązywanego zadania. Jest specyfikowany zbiór obiektów, które mogą stanowić elementy rozwiązania problemu projektowego (zwane obiektami projektowymi) oraz zbiór relacji, jakie mogą zachodzić pomiędzy tymi obiektami.

Relacje te umożliwiają projektantowi wyrażanie faktów, dotyczących zależności pomiędzy obiektami projektowymi i tworzenie syntaktycznego opisu projektu.



Rys. 2.3. Wieża przesyłowa: a) widok, b) diagram projektowy
Fig. 2.3. A transmission tower: a) a view, b) a design diagram

Ponieważ relacje w strukturze projektu są zwykle określane nie pomiędzy całymi komponentami projektowanego artefaktu, ale pomiędzy ich częściami składowymi, rozróżniać będziemy podstawowe obiekty projektowe odpowiadające komponentom rozwiązania i aktywne obiekty projektowe odpowiadające fragmentom tych komponentów. Zbiór obiektów aktywnych zawiera składowe obiektów podstawowych istotne z punktu widzenia relacji zachodzących pomiędzy częściami składowymi projektu. Jest więc definiowane odwzorowanie przypisujące obiektom podstawowym podzbiory obiektów aktywnych.

Niech $O = O_C \cup O_R$ będzie skończonym, niepustym zbiorem elementów zwanych *obiektami projektowymi*, składającym się ze zbioru *obiektów podstawowych* (O_C) odpowiadających komponentom rozwiązania i zbioru *obiektów aktywnych* (O_R). Funkcja *part*: $O_C \rightarrow 2^{O_R}$, przypisuje każdemu obiektowi $o \in O_C$ zbiór obiektów aktywnych. Uporządkowane elementy k-elementowego zbioru *part*(o), $1 \le k \le n$, tworzą ciąg obiektów aktywnych. Obiekt z O_R będący i-tym elementem uporządkowanego ciągu $\{o_1,\ldots, o_k\}$ obiektów aktywnych przypisanych obiektowi o będzie oznaczany przez $o.i$.

Obiekty projektowe są klasyfikowane za pomocą wyspecyfikowanego zbioru typów z określoną taksonomią. Typy wyznaczają klasy obiektów w taki sposób, że klasy te mogą się w sobie zawierać. Na przykład, wykorzystywany w projektowaniu pomieszczeń obiekt typu „ściana_z_przejściem” (*wall_pass*) jest także typu „ściana” (*wall*). Ponadto, obiektom projektowym są przypisywane zbiory atrybutów pozwalających reprezentować własności obiektów, takie jak kształt, rozmiar, lokalizacja, kolor, materiał czy tekstura.

Definicja 2.2:

Konceptualizacja dziedziny projektowania wizualnego (konceptualizacja projektowa) jest szóstką $\mathcal{C} = (O, O_R, A, Att, Z, R)$, gdzie:

- $O = O_C \cup O_R$ jest niepustym zbiorem złożonym z *podstawowych* i *aktywnych obiektów projektowych*,
- A jest skończonym zbiorem *atrybutów*, gdzie Att: $O \rightarrow 2^A$ jest funkcją przypisującą zbiory atrybutów obiektom projektowym,
- Z jest zbiorem typów z określoną taksonomią. Typy te klasyfikują obiekty z O za pomocą relacji zdefiniowanej na $O \times Z$. Jeżeli obiekt o jest w relacji z typem z, to mówimy, że obiekt o jest *typu* z (ozn. $o \in z$),
- R jest zbiorem wieloargumentowych relacji, zwanych *faktami*, postaci $r \subseteq (O_R)^n$, $n \geq 1$. □

Przypisanie obiektom występującym w definicji konceptualizacji projektowej elementów z określonej dziedziny zastosowań projektowania pozwala wyspecyfikować relacje zachodzące między tymi obiektami. W ten sposób otrzymujemy konceptualizację zorientowaną problemowo, zwaną konceptualizacją zadaniową, np. służącą projektowaniu rozkładów pomieszczeń, mebli czy kratownic.

Określenie konceptualizacji projektowej zorientowanej problemowo determinuje wszystkie własności syntaktyczne obiektów, jakie będzie można wyrazić w danym języku projektowania. Konceptualizacja zadaniowa wraz z wymaganiami i kryteriami projektowymi pozwalają zdefiniować syntaktyczne opisy wszystkich projektów z rozważanej dziedziny zastosowań. Syntaktyczny opis projektu nazywany strukturą projektu jest zbiorem faktów opisujących relacje zachodzące pomiędzy elementami wybranego zbioru atrybutowanych obiektów.

Definicja 2.3:

Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową.

Niech $O \supseteq Q = Q_C \cup Q_R$ będzie skończonym podzbiorem obiektów projektowych złożonym z wybranego zbioru obiektów podstawowych $Q_C \subseteq O_C$ i zbioru wyznaczonych dla nich obiektów aktywnych $Q_R \subseteq O_R$, tzn. $Q_R = \{o \in O_R \mid \exists\, o' \in Q_C, o \in part(o')\}$.

Struktura projektu zdefiniowana dla Q jest postaci $Str = (Q, Att_Q, R_Q)$, gdzie Att_Q jest funkcją atrybutowania Att zawężoną do zbioru Q, a $R_Q \subseteq R$ jest skończonym podzbiorem relacji z R zawężonych do obiektów aktywnych z $Q_R \subset Q$. □

Struktury projektów będą następnie reprezentowane w obszarze wizualnym za pomocą uogólnionych dwu- lub trójwymiarowych diagramów projektowych.

W rozważanych w pracy przykładach dotyczących projektowania, w celu uproszczenia wykorzystywanej notacji, obiekty będą oznaczane odpowiednimi skrótami. Ponadto, ponie-

waż przykłady są ilustrowane rysunkami pochodzącymi z prototypowych programów, w których występuje terminologia angielska, obiekty, ich typy, atrybuty, a także relacje zachodzące pomiędzy obiektami i operacje wykonywane na obiektach zachowują nazwy angielskie. Jeżeli wśród argumentów relacji znajdują się wszystkie obiekty aktywne przypisane danemu obiektowi *o* funkcją *part*, wówczas dla uproszczenia notacji ciąg *o*.1,…,*o.n*, gdzie |*part*(*o*)| = *n*, jest zastępowany przez *o*.

Przykład 2.4: rozkłady pomieszczeń

Lista skróconych oznaczeń obiektów, którym odpowiadają komponenty diagramów reprezentujących rozkłady pomieszczeń, jest przedstawiona w tabeli 2.1.

Tabela 2.1
Lista skrótów oznaczających obiekty reprezentowane w diagramach rozkładów pomieszczeń

Obiekt	Skrót	Obiekt	Skrót
obszar mieszkalny	*Ap*	obszar rekreacyjny	*La*
obszar sypialny	*Sa*	salon	*Lr*
kuchnia	*Kt*	sypialnia	*Be*
łazienka	*Ba*	hol	*Hl*
przedsionek	*En*	toaleta	*Wc*
garaż	*Gr*	jadalnia	*Dr*

Konceptualizacja dla języka służącego projektowaniu rozkładów pomieszczeń jest określona w następujący sposób. Zbiór *O* zawiera obiekty reprezentujące pomieszczenia, np. sypialnia, kuchnia, jadalnia, salon, łazienka, toaleta, przedsionek, hol, garaż (*bedroom*, *kitchen*, *dining-room*, *living-room*, *bathroom*, *toilet*, *entrance*, *hall*, *garage*), obiekty reprezentujące obszary, np. obszar użytkowy, mieszkalny, rekreacyjny, komunikacyjny, sypialny (*estate*, *apartment*, *living-part*, *communication-part*, *sleeping-part*) oraz obiekty aktywne, reprezentujące ściany (*wall*) i ściany z przejściem (*wall_with_passage*). Zbiór *Z* = {*room*, *zone*, *wall*, *wall_ pass*} zawiera typy klasyfikujące obiekty z *O* jako pomieszczenia, obszary, ściany i ściany z przejściem. Taksonomia zbioru *Z* jest określona w taki sposób, że wszystkie obiekty typu *wall_ pass* są także typu *wall*, a wszystkie obiekty typu *room* są także typu *zone*. Funkcja *part* wyznaczająca obiekty aktywne, każdemu obiektowi typu *zone* lub *room* przypisuje wszystkie obiekty typu *wall* reprezentujące ściany należące do tego obszaru lub pomieszczenia.

Obiektom typu *zone* lub *room* są przypisane atrybuty *area* i *position* określające powierzchnię i lokalizację tych obiektów. Obiektom typu *wall* są przypisane atrybuty *length*, *position, order, orientation*, *window_number* określające odpowiednio długość, lokalizację, kolejność, orientację ścian oraz liczbę ich okien, a obiektom typu *wall_pass* jest przypisany dodatkowo atrybut *door_number* określający liczbę drzwi.

Zbiór relacji $R = \{acc, adj, cnts\}$ zawiera dwie relacje *acc* i *adj*, określające dostępność i przyległość pomiędzy obszarami lub pomieszczeniami poprzez odpowiednie ich ściany oraz wieloargumentową relację *cnts*, gdzie $cnts(o_1, o_2,\ldots, o_n)$ oznacza, że obiekt o_1 zawiera obiekty $o_2,\ldots, o_n$ (ciągi wszystkich obiektów aktywnych przypisanych obiektom $o_1, o_2,\ldots, o_n$ zostały tu zastąpione tymi obiektami).

Struktura projektu odpowiadającego diagramowi przedstawionemu na rys. 2.1b składa się z ośmiu obiektów typu *room* i 32 obiektów typu *wall* (reprezentujących cztery ściany każdego z pomieszczeń). Na rysunku odcinki reprezentujące ściany wspólne dla 2 różnych pomieszczeń pokrywają się ze sobą, jednakże są one osobno numerowane względem każdego z pomieszczeń. Jedenaście spośród obiektów typu *wall* jest obiektami typu *wall_pass*, przy czym dla obiektów etykietowanych *Lr*.2, *Hl*.4 i *Hl*.3 (zgodnie z notacją dla obiektów aktywnych i tabelą 1), które reprezentują ściany z drzwiami, wartość atrybutu *door_number* wynosi 2. Pomiędzy obiektami typu *wall* zachodzi siedem relacji określających dostępność pomiędzy pomieszczeniami: *acc*(*Lr*.3, *Kt*.1), *acc*(*Lr*.2, *Be*.4), *acc*(*Lr*.2, *Hl*.4), *acc*(*Kt*.2, *Hl*.4), *acc*(*Be*.2, *Ba*.4), *acc*(*Hl*.3, *En*.1), *acc*(*Hl*.3, *Wc*.1) oraz siedem relacji przyległości pomiędzy pomieszczeniami: *adj*(*Be*.3, *Hl*.1), *adj*(*Be*.2, *Gr*.4), *adj*(*Ba*.3, *Gr*.1), *adj*(*Hl*.2, *Gr*.4), *adj*(*Wc*.2, *Gr*.4), *adj*(*En*.2, *Wc*.4), *adj*(*Kt*.2, *En*.4). W przypadku relacji *acc* i *adj*, które są symetryczne, tylko jedna relacja z każdej pary $r(o_1, o_2)$, $r(o_2, o_1)$, gdzie *r* oznacza *acc* lub *adj*, jest brana pod uwagę. ◊

Przykład 2.5: krzesła

Lista skróconych oznaczeń obiektów, którym odpowiadają komponenty diagramów reprezentujących krzesła, jest przedstawiona w tabeli 2.2.

Tabela 2.2

Lista skrótów oznaczających obiekty reprezentowane w diagramach krzeseł

Obiekt	Skrót
siedzisko	*St*
podstawa	*Bs*
i-ta noga	L_i
oparcie	*Bk*
j-ty element oparcia	Bk_j
k-ty element poręczy	P_k

Konceptualizacja dla języka służącego projektowaniu krzeseł jest określona w następujący sposób. Zbiór obiektów *O* zawiera siedzisko, oparcie, podstawę oraz elementy oparcia, poręczy, nogi krzeseł oraz punkty połączenia będące obiektami aktywnymi, indeksowane kolejnymi liczbami naturalnymi. Zbiór typów klasyfikujących obiekty jest postaci $Z = \{seat, back, back\_element, arm\_element, base, leg, connection\_point\}$. Taksonomia zbioru *Z* jest określona w taki sposób, że typ *leg* zawiera się w typie *base*, a typ *back_element* zawiera się

w typie *back*. Każdemu obiektowi typu *seat*, *back*, *back_element*, *arm_element*, *base* lub *leg* są przypisane będące ich fragmentami obiekty typu *connection_point*. Obiektom typu *connection_point* są przypisane atrybuty *position* i *order*, określające lokalizację punktów połączenia obiektów oraz ich kolejność. Pozostałym obiektom są przypisane atrybuty *width*, *height*, *depth*, *colour*, *material*, *texture* określające ich rozmiary i wygląd. Zbiór $R = \{adj, cnts\}$ zawiera relację *adj* określającą przyległość pomiędzy punktami połączenia obiektów reprezentujących elementy krzesła i relację zawierania *cnts* zachodzącą pomiędzy podstawą krzesła i jego nogami, oraz pomiędzy oparciem i jego elementami składowymi.

Struktura projektu krzesła odpowiadającego diagramowi przedstawionemu na rys. 2.2b zawiera jeden obiekt typu *seat* z sześcioma obiektami typu *connction_point*, trzy obiekty typu *back_element* każdy z dwoma obiektami typu *connction_point* i cztery obiekty typu *leg* każdy z jednym obiektem typu *connction_point*. Struktura ta zawiera osiem relacji przyległości pomiędzy punktami połączeń odpowiednich elementów. ◊

Przykład 2.6: wieże przesyłowe

Lista skróconych oznaczeń obiektów, którym odpowiadają komponenty diagramów reprezentujących kratowe wieże przesyłowe, jest przedstawiona w tabeli 2.3.

Tabela 2.3
Lista skrótów oznaczających obiekty reprezentowane w diagramach wież przesyłowych

Obiekt	Skrót
segment dolny	Sg_b
segment środkowy	Sg_m
segment górny	Sg_t
i-te skratowanie	T_i
j-ty element skratowania	Te_j
k-ty izolator	I_k

Zbiór obiektów *O* zawiera segment dolny, segment środkowy, segment górny, skratowania, elementy skratowań, izolatory oraz będące obiektami aktywnymi węzły skratowań, indeksowane kolejnymi liczbami naturalnymi. Zbiór typów klasyfikujących obiekty jest postaci $Z = \{segment, truss, truss\_panel, truss\_node, insulator\}$. Wszystkie obiekty typu *truss_panel* są też typu *truss*. Funkcja *part* określająca obiekty aktywne, przypisuje segmentom, skratowaniom, elementom skratowań i izolatorom węzły typu *truss_node* reprezentujące punkty zespolenia elementów wieży.

Wszystkim obiektom są przypisane atrybuty *width*, *length* i *position* określające ich rozmiar i lokalizację. Obiektom typu *truss*, *truss_panel* i *insulator* są przypisane ponadto atrybuty *type* i *material*, określające rodzaj obiektu i z jakiego materiału jest on wykonany, a obiektom typu *truss_node* – atrybut *order* określający kolejność węzłów skratowania. Zbiór $R = \{con, cnts\}$ zawiera relację *con*, określającą zespolenie pomiędzy węzłami elementów

wieży, oraz relację zawierania *cnts* zachodzącą pomiędzy segmentami wieży lub skratowaniami i ich elementami składowymi.

Struktura projektu wieży przesyłowej odpowiadającego diagramowi przedstawionemu na rys. 2.3b zawiera sześć obiektów typu *truss_panel* reprezentujących elementy skratowania, każdy z czterema obiektami typu *truss_node* reprezentującymi węzły tych elementów, siedem obiektów typu *truss* reprezentujących poziome skratowania, każdy z dwoma obiektami typu *truss_node* i cztery obiekty typu *insulator* każdy z jednym obiektem typu *truss_node*. Struktura ta zawiera szesnaście relacji zespolenia pomiędzy węzłami elementów wieży. ◊

W trakcie procesu projektowego obiekty projektowe są przekształcane za pomocą operacji projektowych (np. podział lub scalanie). Operacje te jak również zmiany relacji między obiektami struktury projektu, które są wynikiem zastosowania tych operacji, są specyfikowane przez syntaktyczne reguły projektowe. Reguły te określają także ograniczenia dotyczące obiektów struktury, które muszą być spełnione, aby dana operacja mogła być zastosowana.

Definicja 2.4:
Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową.

Syntaktyczna reguła projektowa zdefiniowana nad $\mathcal{C}$ jest czwórką $u = (t, \delta, \alpha, \beta)$, gdzie:

- $t \in T$, gdzie T jest zbiorem odwzorowań, zwanych *operacjami projektowymi*, postaci $t: O^n \to O^m$, $n, m \geq 1$,
- δ jest zbiorem ograniczeń dotyczących obiektów z O, które warunkują zastosowanie operacji t,
- $\alpha, \beta \subseteq R$, α jest zbiorem faktów, które zachodzą przed wykonaniem t, a β jest zbiorem faktów, które zachodzą po wykonaniu t. □

Przykład 2.7: rozkłady pomieszczeń
Przykładowe operacje projektowe na obiektach konceptualizacji służącej projektowaniu rozkładów pomieszczeń to $divide(o_1, o_1.1,\ldots, o_1.m) = (o_1, o_2,\ldots, o_n)$ i $merge(o_1, o_2,\ldots, o_n) = (o_1, o_1.1,\ldots, o_1.m)$, gdzie o_1 jest typu *zone* lub *room*, $o_1.1,\ldots, o_1.m$ jest sekwencją obiektów typu *wall* reprezentujących ściany obiektu o_1, a $o_2,\ldots, o_n$ są obiektami typu *zone*, *room* lub *wall*. Te dwie operacje pozwalają projektantowi podzielić obszar lub pomieszczenie na kilka innych obszarów lub pomieszczeń oraz usunąć istniejący podział danego obszaru lub pomieszczenia.

Warunek δ reguły projektowej dla operacji *divide* mówi, że dany obiekt może być podzielony, jeśli nie zawiera innych obiektów typu *zone* lub *room*. Zbiór α faktów, które ulegają zmianie w wyniku zastosowania operacji *divide*, zawiera relacje *acc* i *adj* zachodzące dla

obiektów reprezentujących ściany obiektu o_1. Zbiór β faktów, które zachodzą po wykonaniu tej operacji, zawiera relacje *acc* i *adj* zachodzące dla obiektów występujących w sekwencji $o_2,\ldots, o_n$ i reprezentujących ściany oraz relację *cnts* zachodzącą pomiędzy obiektem o_1 i obiektami typu *zone* lub *room* występującymi w sekwencji $o_2,\ldots, o_n$. Warunek δ reguły projektowej dla operacji *merge* mówi, że dany obiekt może być scalony, jeśli zawiera inne obiekty. Zbiór α faktów, które ulegają zmianie w wyniku zastosowania operacji *merge*, zawiera relację *cnts* zachodzącą pomiędzy obiektem o_1 i obiektami typu *zone* lub *room* występującymi w sekwencji $o_2,\ldots, o_n$ oraz relacje *acc* i *adj* zachodzące dla obiektów występujących w sekwencji $o_2,\ldots, o_n$ i reprezentujących ściany. Zbiór β w przypadku tej operacji zawiera relacje *acc* i *adj* zachodzące dla obiektów reprezentujących ściany obiektu o_1. ◊

Przykład 2.8: krzesła

W przypadku projektowania krzeseł zbiór T zawiera między innymi operacje $divide(o_1, o_1.1,\ldots, o_1.m) = (o_1, o_2,\ldots, o_n)$ i $merge(o_1, o_2,\ldots, o_n) = (o_1, o_1.1,\ldots, o_1.m)$, gdzie: o_1 jest oparciem lub podstawą, $o_1.1,\ldots, o_1.m$ jest sekwencją punktów połączeń obiektu o_1, a $o_2,\ldots, o_n$ są odpowiednio elementami oparcia lub nogami krzesła i ich punktami połączeń. Operacja *divide* jest stosowana bezwarunkowo, a operacja *merge* tylko jeśli obiekt, będący jej pierwszym argumentem, zawiera inne obiekty. Zbiór α faktów, które ulegają zmianie w wyniku zastosowania operacji *divide*, zawiera relacje *adj* zachodzące dla punktów połączeń obiektu o_1, a zbiór β faktów, które zachodzą po wykonaniu tej operacji, zawiera relacje *adj* zachodzące dla punktów połączeń obiektów występujących w sekwencji $o_2,\ldots, o_n$ i relację *cnts* zachodzącą pomiędzy obiektem o_1 oraz obiektami występującymi w sekwencji $o_2,\ldots, o_n$ i niebędącymi punktami połączeń. Zbiór α faktów, które ulegają zmianie w wyniku zastosowania operacji *merge*, zawiera relację *cnts* zachodzącą pomiędzy obiektem o_1 oraz obiektami występującymi w sekwencji $o_2,\ldots, o_n$ i niebędącymi punktami połączeń, a także relacje *adj* zachodzące dla punktów połączeń obiektów występujących w sekwencji $o_2,\ldots, o_n$, a zbiór β zawiera relacje *adj* zachodzące dla punktów połączeń obiektu o_1 po wykonaniu tej operacji.

◊

Przykład 2.9: wieże przesyłowe

Operacje wykonywane na obiektach wieży przesyłowej umożliwiają podział segmentów na skratowania, elementy skratowań i izolatory oraz podział skratowań na tworzące je elementy. Operacja scalania obiektów daje możliwość usunięcia podziału dowolnego segmentu wieży.

◊

Proces projektowy jest zwykle rozpoczynany od początkowej struktury projektu, która jest strukturą zawierającą co najmniej jeden obiekt projektowy. Następnie do obiektów struktury są stosowane operacje projektowe modyfikujące strukturę projektu, zgodnie z syntak-

tycznymi regułami projektowymi. Reguły te wraz z początkową strukturą projektu tworzą projektowy system generacyjny.

Definicja 2.5:
Projektowym systemem generacyjnym nazywamy parę $S_G = (U, Str_I)$, gdzie:

- U jest skończonym, niepustym zbiorem syntaktycznych reguł projektowych postaci $u = (t, \delta, \alpha, \beta)$ zdefiniowanych nad konceptualizacją projektową $\mathcal{C} = (O, O_R, A, Att, Z, R)$,
- $Str_I = (Q_I, Att_{Q_I}, R_{Q_I})$ jest **początkową strukturą projektu** zdefiniowaną dla zbioru Q_I zawierającego co najmniej jeden obiekt projektowy z O. □

Definicja 2.6:
Niech S_G będzie projektowym systemem generacyjnym, którego reguły są zdefiniowane nad konceptualizacją projektową $\mathcal{C} = (O, O_R, A, Att, Z, R)$.

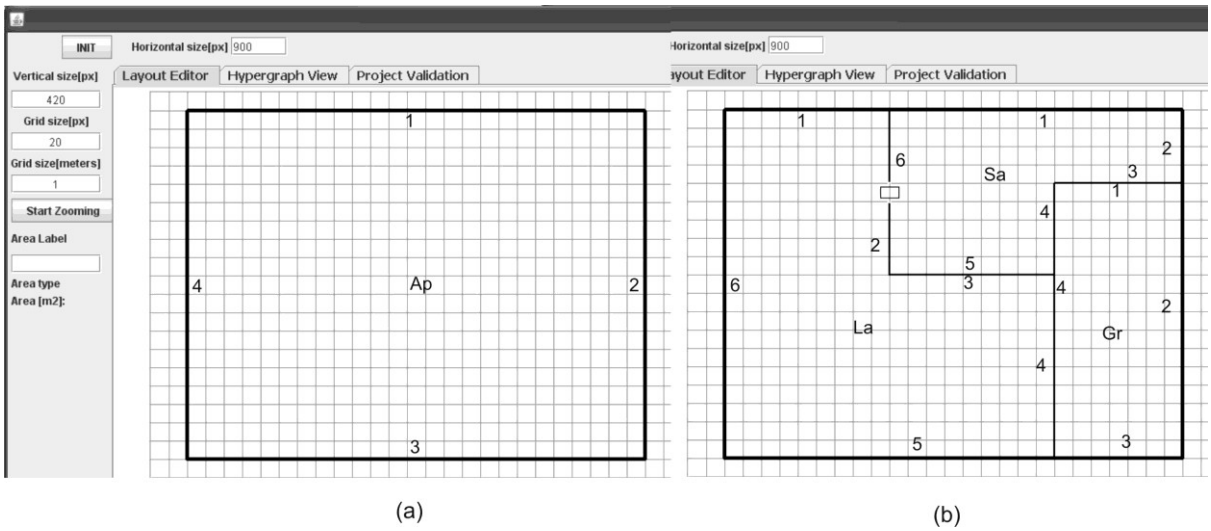
Wynikiem **zastosowania reguły projektowej** $u = (t, \delta, \alpha, \beta) \in U$, gdzie t jest operacją projektową postaci $t(o_{i_1}, o_{i_2}, \ldots, o_{i_n}) = (o_{j_1}, o_{j_2}, \ldots, o_{j_m})$, **do struktury projektu** $Str = (Q, Att_Q, R_Q)$, której obiekty spełniają warunki określone przez δ, jest struktura projektu $Str' = (Q', Att_{Q'}, R_{Q'})$ (ozn. $Str \Rightarrow Str'$), gdzie:

- $Q' = (Q - Q_1) \cup Q_2$, gdzie $Q_1 = \{o_{i_1}, o_{i_2}, \ldots, o_{i_n}\}$, $Q_2 = \{o_{j_1}, o_{j_2}, \ldots, o_{j_m}\}$,
- $Att_{Q'}$ jest funkcją atrybutowania Att zawężoną do zbioru Q',
- $R_{Q'} = (R_Q - \alpha) \cup \beta$. □

Przykład 2.10: rozkłady pomieszczeń
Początkowej strukturze projektu, od której jest rozpoczynane projektowanie rozkładu pomieszczeń mieszkania, odpowiada diagram z rys. 2.4a. Struktura ta składa się z obiektu *apartment* i czterech przypisanych mu obiektów typu *wall* reprezentujących ograniczające go ściany. W następnym kroku projektowym początkowy obiekt *apartment* jest dzielony operacją *divide*(*Ap*, *Ap*.1, *Ap*.2, *Ap*.3, *Ap*.4) na obszar rekreacyjny, sypialny i garaż. W wyniku tej operacji do struktury projektu zostały dodane trzy nowe obiekty: *living-part*, *sleeping-part* i *garage* oraz szesnaście nowych obiektów odpowiadających ścianom otaczającym nowe obszary, wśród których znajduje się jeden obiekt typu *wall_pass*. Do struktury projektu dodane zostały także relacje *acc*(*La*.2, *Sa*.6), *adj*(*La*.3, *Sa*.5), *adj*(*La*.4, *Gr*.4), *adj*(*Sa*.3, *Gr*.1) i *adj*(*Sa*.4, *Gr*.4). Ponadto, po wykonaniu tej operacji zachodzi relacja *cnts*(*Ap*, *La*, *Sa*, *Gr*). Diagram odpowiadający nowej strukturze projektu jest pokazany na rys. 2.4b. ◊

Przykład 2.11: krzesła
Strukturze projektu, od której jest rozpoczynane projektowanie krzesła, odpowiada diagram przedstawiony na rys. 2.5a. Struktura ta składa się z trzech atrybutowanych obiektów: pod-

stawa (*Bs*), siedzisko (*St*), oparcie (*Bk*), 12 obiektów reprezentujących ich punkty połączeń oraz relacji *adj*(*Bk*.1, *St*.2), *adj*(*Bk*.2, *St*.1), *adj*(*Bs*.1, *St*.6), *adj*(*Bs*.2, *St*.5), *adj*(*Bk*.3, *St*.4) i *adj*(*Bs*.4, *St*.3). ◊



Rys. 2.4. Diagramy projektowe w programie *HSSDR*: a) początkowy diagram projektowy, b) diagram otrzymany po operacji podziału apartamentu

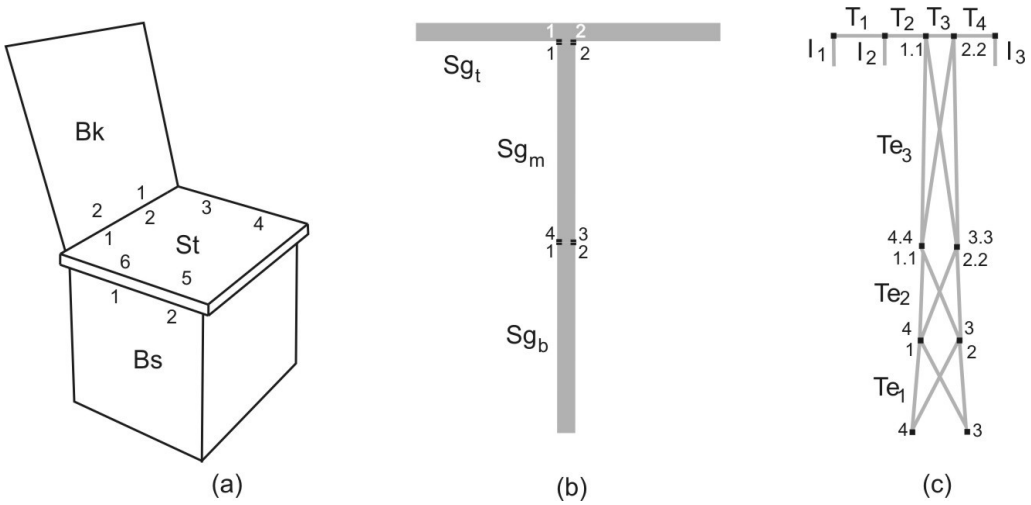
Fig. 2.4. Design diagrams in *HSSDR* program: a) an initial design diagram, b) the diagram obtained after the apartment division operation

Przykład 2.12: wieże przesyłowe

Diagram przedstawiony na rys. 2.5b odpowiada początkowej strukturze projektu wieży przesyłowej. Struktura ta składa się z trzech obiektów odpowiadających segmentowi dolnemu (Sg_b), środkowemu (Sg_m) i górnemu (Sg_t), 8 obiektów reprezentujących węzły, w których skratowania tych segmentów będą się łączyły ze sobą, dwóch relacji zespolenia pomiędzy węzłami segmentu dolnego i środkowego oraz dwóch relacji zespolenia pomiędzy węzłami segmentu środkowego i górnego. W kolejnych krokach projektowych segmenty te będą dzielone na skratowania i ich poszczególne elementy. Diagram projektowy, odpowiadający strukturze projektu otrzymanej w wyniku podziału segmentu dolnego na dwa elementy skratowania (Te_1, Te_2), zastąpienia segmentu środkowego jednoelementowym skratowaniem (Te_3), oraz podziału segmentu górnego na cztery skratowania (T_1-T_4) i trzy izolatory (I_1-I_3), jest przedstawiony na rys. 2.5c. Otrzymana struktura zawiera także obiekty odpowiadające węzłom skratowań, w których elementy są łączone oraz 9 relacji opisujących te połączenia. Na rys. 2.5c są przedstawione tylko numery węzłów skratowań należących do segmentu dolnego i środkowego. ◊

Struktura danego projektu jest *syntaktycznie poprawna*, jeśli wszystkie fakty dotyczące relacji pomiędzy obiektami tej struktury mogą być otrzymane poprzez stosowanie kolejnych operacji na obiektach oraz zastępowanie istniejących faktów nowymi faktami w sposób określony przez reguły syntaktyczne. Zbiór wszystkich struktur projektów, które da się otrzymać

z danej struktury początkowej poprzez zastosowanie reguł projektowych danego systemu generacyjnego, jest nazywany syntaktyką języka projektowania.



Rys. 2.5. Diagramy projektowe: a) początkowy diagram dla krzesła, b) początkowy diagram dla wieży przesyłowej, c) diagram odpowiadający strukturze zaprojektowanej wieży
Fig. 2.5. Design diagrams: a) an initial diagram of a chair, b) an initial diagram of a transmission tower, c) the diagram corresponding to the structure of the designed tower

Definicja 2.7:

Niech Str_I oznacza początkową strukturę projektu, S_G będzie systemem generacyjnym, a „$\Rightarrow^*$” – oznacza przechodnie domknięcie zastosowania reguły projektowej z S_G („$\Rightarrow$”).

Syntaktyka *Snt* diagramowego języka projektowania jest zbiorem syntaktycznie poprawnych struktur projektów, tzn. $Snt = \{Str \mid Str_I \Rightarrow^* Str\}$. □

W zależności od dziedziny projektowania, dla której jest określona konceptualizacja, struktury projektów tworzą syntaktykę języków projektowania różnych typów.

2.3.3. Realizacja syntaktyki diagramowego języka projektowania

W celu stworzenia diagramu reprezentującego rozwiązanie projektowe należy nadać znaczenie geometryczne zarówno obiektom, jak i relacjom zachodzącym między obiektami występującymi w strukturze projektu. Służy temu rodzina odwzorowań, zwana realizacją syntaktyki *DJP*, która odwzorowuje struktury projektów w dziedzinę diagramów projektowych. Dziedzina diagramów projektowych zawiera słownik złożony z podstawowych elementów wizualnych odpowiadających obiektom określonym w konceptualizacji projektowej, zbiór dopuszczalnych transformacji dla elementów słownika, zbiór relacji przestrzennych pomiędzy komponentami diagramu otrzymanymi poprzez transformowanie elementów słownika oraz funkcję atrybutowania komponentów diagramu.

Definicja 2.8:

Dziedzina diagramów projektowych jest szóstką $DG = (S, F, S', A', Att', X)$, gdzie:

- S jest skończonym zbiorem podstawowych elementów wizualnych będących ograniczonymi podzbiorami $\mathfrak{R}^n$,
- F jest zbiorem dopuszczalnych transformacji postaci $f: \mathfrak{R}^n \to \mathfrak{R}^n$,
- $S' \subseteq \{f(s) \mid f \in F,\ s \in S\}$ jest zbiorem przetransformowanych elementów z S, zwanych *komponentami diagramów*,
- A' jest zbiorem atrybutów określonych dla dziedziny DG, a $Att': S' \to 2^{A'}$ jest funkcją przypisującą zbiory atrybutów komponentom z S',
- X jest zbiorem wieloargumentowych relacji przestrzennych pomiędzy komponentami diagramów postaci $x \subseteq (S')^n$, $n \geq 1$. □

Przykład 2.13: rozkłady pomieszczeń

W przypadku projektowania rozkładów pomieszczeń słownik S zawiera wielokąty odpowiadające obszarom i pomieszczeniom, odcinki oraz odcinki z umieszczonymi na nich prostokątami odpowiadające ścianom i ścianom z przejściem. Zbiór F dopuszczalnych transformacji składa się z przesunięć, obrotów i skalowania. Funkcja Att' przypisuje przetransformowanym wielokątom atrybuty określające ich rozmiary i pozycje. Atrybuty określające długość, orientację, lokalizację, kolejność, liczbę drzwi i okien są przypisywane przetransformowanym odcinkom występującym w diagramach.

Zbiór relacji przestrzennych $X = \{sq\_line, c\_line, inside\}$ zawiera dwie relacje binarne – *sq_line* i *c_line*, z których pierwsza jest spełniona, jeśli sąsiednie wielokąty posiadają wspólny odcinek z umieszczonym na nim prostokątem, a druga – jeśli mają one wspólny odcinek bez narysowanego na nim prostokąta. *Inside* jest wieloargumentową relacją zachodzącą jeśli pierwszy argument jest wielokątem zawierającym wewnątrz wszystkie wielokąty będące pozostałymi argumentami. ◊

Przykład 2.14: krzesła

W przypadku projektowania krzeseł słownik zawiera trójwymiarowe bryły odpowiadające poszczególnym ich elementom. Fragmenty tych brył reprezentują punkty połączeń pomiędzy częściami krzesła. Zbiór F dopuszczalnych transformacji składa się z przesunięć, obrotów i skalowania. Funkcja Att' przypisuje przetransformowanym bryłom atrybuty *width*, *height*, *depth* określające ich rozmiary oraz *colour*, *material*, *texture*, których wartości specyfikują wygląd brył. Odpowiednim fragmentom brył są przypisywane atrybuty *position* i *order*, określające lokalizację punktów połączenia brył oraz ich kolejność. Zbiór relacji przestrzennych zawiera relacje *touch* i *inside* oznaczające przyleganie do siebie fragmentów brył i zawieranie się brył w sobie. ◊

Przykład 2.15: wieże przesyłowe

W przypadku projektowania wież przesyłowych słownik zawiera kształty odpowiadające różnym typom elementów skratowania, oraz odcinki odpowiadające skratowaniom poziomym i izolatorom. Odpowiednie punkty tych kształtów i końce odcinków reprezentują węzły, w których skratowania są zespalane lub łączone z izolatorami. Zbiór F dopuszczalnych transformacji składa się z przesunięć, obrotów i skalowania. Funkcja *Att*' przypisuje przetransformowanym kształtom i odcinkom atrybuty, określające ich rozmiary, lokalizację, typy oraz materiał, z jakiego są zrobione. Relacja przestrzenna *touch* oznacza stykanie się punktów reprezentujących węzły wieży. ◊

Rodzina odwzorowań będących realizacją syntaktyki *DJP* składa się z odwzorowań określonych dla wszystkich struktur, stanowiących elementy tej syntaktyki. Realizacja struktury projektu przypisuje obiektom tej struktury odpowiednie elementy słownika wraz z transformacjami, jakim powinny one zostać poddane, aby zachodziły między nimi relacje przestrzenne obrazujące relacje występujące w strukturze projektu. Realizacja struktury projektu jest definiowana na podstawie semantycznej wiedzy projektowej, która zawiera wymagania i ograniczenia projektowe wykorzystywane do sterowania procesem projektowym w celu otrzymania diagramów zgodnych z zamierzoną interpretacją.

Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową, a zbiór *Snt* złożony ze struktur postaci $Str = (Q, Att_Q, R_Q)$ będzie syntaktyką języka *DJP*.

Niech $DG = (S, F, S', A', Att', X)$ będzie dziedziną diagramów projektowych.

Definicja 2.9:

Realizacją syntaktyki *Snt* języka *DJP* nazywamy rodzinę odwzorowań $\tau_{Str} = (\tau^1_{Str}, \tau^2_{Str})$, gdzie:

- $\tau^1_{Str} : Q \times Att_Q(Q) \to S \times F$ przypisuje każdemu atrybutowanemu obiektowi z Q parę (s, f) złożoną z podstawowego elementu wizualnego i określonej dla niego transformacji,
- $\tau^2_{Str} : R_Q \to X$ przypisuje relację przestrzenną każdej relacji ze struktury projektu, w taki sposób, że $\forall\, r \in R_Q \;\; r(o_1,\ldots, o_n) \Rightarrow x(f_1(s_1),\ldots,f_n(s_n))$, gdzie $\tau^1_{Str}(o_j, Att_Q(o_j)) = (s_j, f_j)$ dla $1 \le j \le n$ oraz $\tau^2_{Str}(r) = x$. □

Przyjmuje się, że komponenty diagramu odpowiadające obiektom aktywnym posiadają część wspólną z komponentami odpowiadającymi obiektom projektowym, do których te obiekty aktywne są przypisane.

Realizacja syntaktyki *DJP* określa sposób generowania uogólnionych diagramów projektowych. Uogólniony diagram projektowy zgodny z realizacją struktury projektu jest tworzony poprzez rozmieszczenie komponentów diagramu otrzymanych w wyniku transformacji odpowiednich elementów wizualnych ze słownika. Uogólniony diagram projektowy odpo-

wiadający strukturze projektu *Str* opisującej relacje między obiektami z *Q* i zgodny z realizacją określoną przez τ_{Str}, $Dg = \bigcup_{(s,f)\in\tau^1_{Str}(Q\times Att_Q(Q))} f(s)$, jest sumą wizualnych elementów z *S* przetransformowanych zgodnie z funkcjami z *F* przypisanymi atrybutowanym obiektom z *Q* przez τ^1_{Str}. Relacje przestrzenne zachodzące pomiędzy komponentami diagramu i odpowiadające relacjom występującym w strukturze *Str* są określone przez τ^2_{Str}.

W koncepcyjnej fazie projektowania komponenty diagramów wyrażają podstawowe informacje o projektowanym artefakcie pozwalające określić jego semantykę (np. kształty pokoi i ich wzajemną lokalizację). Podczas realizacji struktury projektu atrybuty przypisane obiektom projektowym mają wpływ na wybór transformacji stosowanych do elementów słownika. Przykładowo, atrybuty określające rozmiary obiektów (jak *lenght*, *width*, *area*) oznaczają potrzebę zastosowania skalowania podstawowych elementów wizualnych, a atrybuty określające lokalizację obiektów (jak *position* czy *orientation*) specyfikują takie transformacje, jak translacje i obroty.

Stworzenie poprzez daną realizację struktury projektu instancji komponentów diagramu nadaje wartości atrybutom tych komponentów, jednocześnie wartościując atrybuty obiektów należących do realizowanej struktury. Pozwala to na sprawdzanie warunków umożliwiających stosowanie syntaktycznych reguł projektowych zmieniających strukturę projektu. Realizacja tworzy więc funkcjonalną wizualizację struktury projektu.

Niech *D*' oznacza zbiór wartości atrybutów należących do zbioru *A*', będącego elementem dziedziny diagramów *DG*, a $S_{Dg} = \{f(s) \in Dg \mid s \in S, f \in F\}$ będzie zbiorem komponentów diagramu *Dg*. Rodzina funkcji częściowych $(Ev'_a)_{a \in A'}$, gdzie Ev'_a: $S_{Dg} \times A' \rightarrow D'_a$ i $D'_a \subseteq D'$ jest zbiorem wartości atrybutu *a*, przypisuje wartości atrybutom komponentów diagramu. Uogólniony diagram *Dg* jest *semantycznie poprawny*, jeśli atrybuty wszystkich jego komponentów (przypisane funkcją *Att*') posiadają wartości, przy których diagram ten spełnia określone kryteria projektowe. Kryteria te w proponowanym systemie wspomagania projektowania będą wyrażone za pomocą formuł logicznych pierwszego rzędu (rozdział 6). Wartości atrybutów komponentów diagramu będą natomiast przechowywane w hipergrafowej strukturze wewnętrznej tego diagramu (rozdział 3).

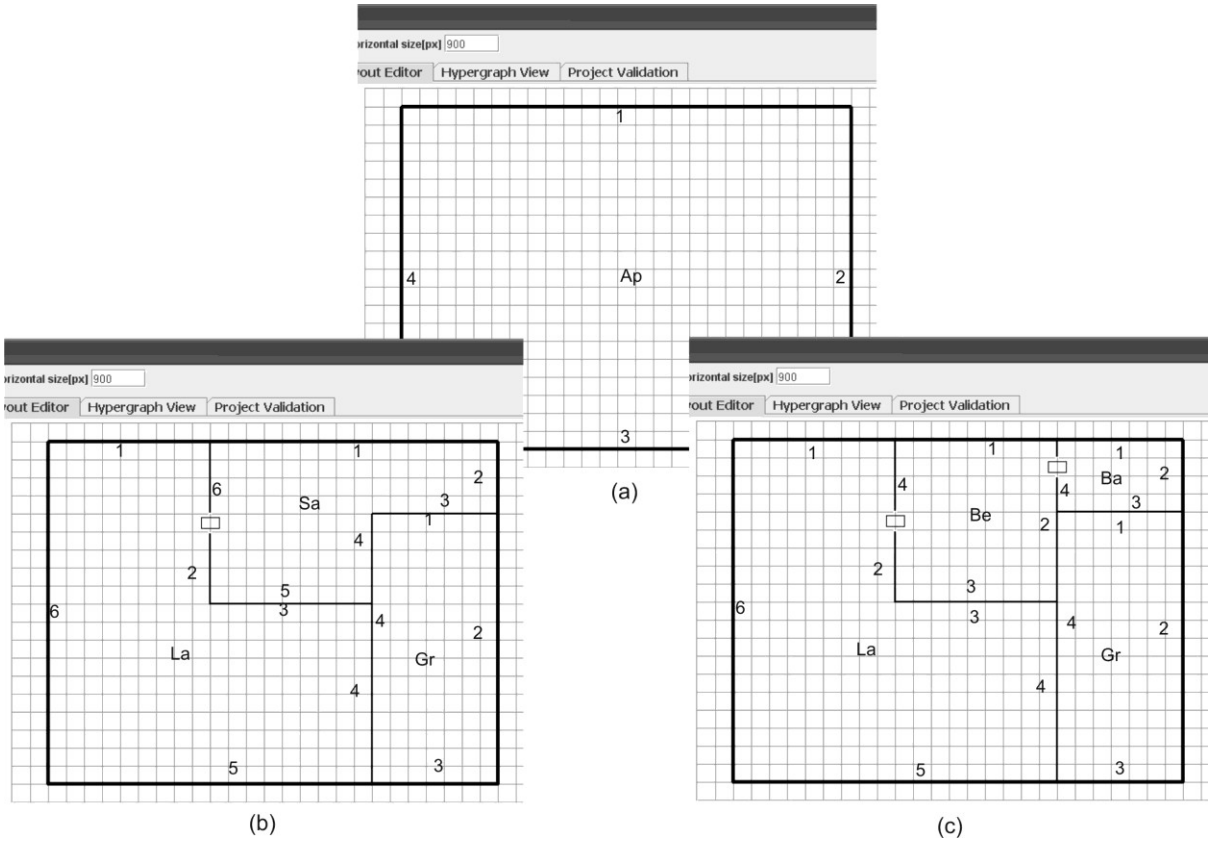
Diagramowy Język Projektowania (*DJP*) składa się ze wszystkich semantycznie poprawnych uogólnionych diagramów projektowych, będących wizualizacjami struktur projektów określonymi przez daną realizację syntaktyki *DJP*.

Przykład 2.16: rozkłady pomieszczeń

W przypadku projektowania rozkładów pomieszczeń realizacja danej struktury projektu przypisuje wielokąty wraz z ich transformacjami obiektom typu *zone* lub *room*, odcinki lub odcinki z umieszczonymi na nich prostokątami wraz z transformacjami odpowiednio obiek-

tom typu *wall* i *wall_pass*. Relacje *sq_line*, *c_line* i *inside* odpowiadają relacjom *acc*, *adj* i *cnts*.

Rozważmy diagramy reprezentujące kolejne kroki projektowania rozkładu pomieszczeń przedstawione na rys. 2.6. Realizacja struktury projektu odpowiadającej diagramowi z rys. 2.6a przypisuje obiektowi *apartment* prostokąt wraz z transformacją, która pozwala przekształcić go w prostokąt pokazany na rys. 2.6a. Wszystkim obiektom typu *wall* są przypisane boki tego prostokąta z transformacjami pozwalającymi każdy odcinek odpowiadający ścianie zawrzeć w odpowiednim boku prostokąta. Rozmiary i lokalizacja przetransformowanego wielokąta i jego boków określają wartości przypisanych im atrybutów *area, position*, *orientation*, *order* i *length*.



Rys. 2.6. Diagramy projektowe w programie *HSSDR*: a) początkowy diagram projektowy, b), c) diagramy reprezentujące kolejne etapy projektowania rozkładu pomieszczeń
Fig. 2.6. Design diagrams in *HSSDR* program: a) an initial design diagram, b), c) diagrams representing successive steps of designing a floor layout

Początkowy obiekt *apartment* został następnie podzielony na obszar rekreacyjny, sypialny i garaż operacją *divide*, która odpowiada narysowaniu kilku nowych odcinków na diagramie (rys. 2.6b). Nowa struktura projektu została omówiona w przykładzie 2.10. Diagram pokazany na rys. 2.6b odpowiadający tej strukturze został otrzymany poprzez przetransformowanie dwóch wielokątów w kształcie litery L przypisanych obiektom *living-part* i *sleeping-*

part oraz prostokąta przypisanego obiektowi *garage*, tak aby zawierały się one w odpowiednich miejscach prostokąta reprezentującego obiekt *apartment*. Lokalizacja i rozmiary wielokątów diagramu określają wartości atrybutów komponentów diagramu.

Następnie obszar sypialny został podzielony na sypialnię i łazienkę za pomocą operacji *divide* (rys. 2.6c). Do struktury projektu zostały dodane obiekty *sypialnia* i *łazienka* reprezentujące te pomieszczenia, obiekty reprezentujące ich ściany oraz relacja *acc*(*Be*.2, *Ba*.4) zachodząca pomiędzy nowymi pomieszczeniami. Ponadto, w nowej strukturze projektu relacja *acc*(*La*.2, *Sa*.6) została zastąpiona relacją *acc*(*La*.2, *Be*.4), relacja *adj*(*La*.3, *Sa*.5) – relacją *adj*(*La*.3, *Be*.3), relacja *adj*(*Sa*.4, *Gr*.4) została zastąpiona relacją *adj*(*Be*.2, *Gr*.4), a relacja *adj*(*Sa*.3, *Gr*.1) relacją *adj*(*Ba*.3, *Gr*.1). Dodatkowo zachodzi relacja *cnts*(*Sa*, *Be*, *Ba*). Diagram przedstawiony na rys. 2.6c został otrzymany poprzez odpowiednie przetransformowanie prostokątów przypisanych nowym obiektom i przypisanie przestrzennej relacji *sq_line* relacji *acc*. ◊

2.4. Podsumowanie

W tym rozdziale zostały omówione języki wizualne i ich zastosowanie w projektowaniu. Zostały przedstawione diagramy projektowe będące wizualizacją koncepcyjnego etapu rozwiązań projektowych i stanowiące podzbiór diagramów wizualnych. W celu charakteryzacji diagramowego języka projektowania składającego się z diagramów projektowych i umożliwiającego dialog między projektantem a systemem komputerowym, została wprowadzona konceptualizacja projektowa. Klasyfikuje ona pojęcia z dziedziny projektowania wizualnego i pozwala wyspecyfikować syntaktyczne reguły przekształcania diagramów. System generacyjny bazujący na tych regułach umożliwia przekształcanie zdefiniowanych tu struktur projektów. Została przedstawiona syntaktyka diagramowego języka projektowania składająca się ze struktur projektów modelowanych artefaktów. Następnie została wyspecyfikowana realizacja syntaktyki określająca semantykę diagramowego języka projektowania. Realizacja nadaje znaczenie obiektom i relacjom struktur projektów, odwzorowując te struktury w zdefiniowaną tu dziedzinę diagramów projektowych. Realizacja syntaktyki umożliwia więc otrzymywanie diagramów projektowych o określonej semantyce. Wykorzystanie diagramowego języka projektowania zostało przedstawione na przykładach projektowania rozkładów pomieszczeń, krzeseł oraz topologii kratowych wież przesyłowych.

W rozdziale 3 struktury projektów będą reprezentowane za pomocą atrybutowanych hierarchicznych hipergrafów, natomiast diagramy projektowe za pomocą wartościowanych hierarchicznych hipergrafów, gdzie atrybuty hipergrafów posiadają konkretne wartości określające semantykę diagramów. Ontologiczne podejście do syntaktyki i semantyki diagramowego języka projektowania zostanie wykorzystane w modelu wnioskowania o poprawności rozwiązań opartym na formułach logicznych pierwszego rzędu, który będzie omówiony w roz-

dziale 6. Model ten pozwala sprawdzać poprawność diagramów projektowych względem określonych wymagań i ograniczeń projektowych z wykorzystaniem wewnętrznej hipergrafowej reprezentacji diagramów.

3. HIERARCHICZNE HIPERGRAFY W PROJEKTOWANIU

Jednym z głównych wymagań projektowych wpływających na etap projektowania koncepcyjnego jest funkcjonalność projektowanego obiektu. W niektórych systemach komputerowych wspomagających projektowanie konceptualne [BoGH99, Szub05] jest stosowana metoda funkcjonalno-strukturalna, gdzie projektowanie rozpoczyna się od reprezentacji wymagań funkcjonalnych i zależności pomiędzy wymaganiami projektowymi za pomocą grafu o wierzchołkach etykietowanych nazwami elementów funkcjonalnych, zwanego grafem funkcyjnym.

Graf funkcyjny stanowi punkt wyjścia dla projektanta i ułatwia ustalenie odpowiedniości pomiędzy funkcjonalnymi i strukturalnymi własnościami projektowanego obiektu. W kolejnych krokach projektowania projektant może stworzyć na podstawie grafu funkcyjnego graf hierarchiczny reprezentujący strukturę projektowanego obiektu, która zostanie zinterpretowana na przykład w postaci diagramu projektowego.

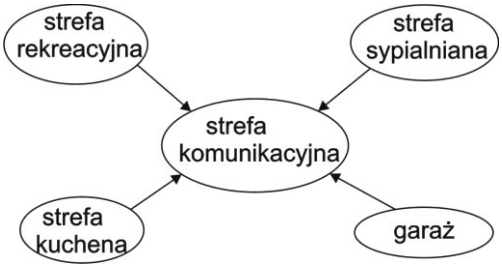
W pracy [BoGH99] został przedstawiony edytor umożliwiający modyfikowanie funkcjonalno-strukturalnego grafu, którego wierzchołkom przypisano kolory reprezentujące rożne wymagania funkcjonalne, za pomocą transformacji grafowych. Operacja *merge* umożliwia łączenie kilku wierzchołków grafu reprezentujących różne funkcje w jeden różnokolorowy wierzchołek, a operacja *split* umożliwia zastąpienie jednego wierzchołka podgrafem złożonym z elementów pełniących tę samą funkcję co zastępowany wierzchołek. Jednakże wykonywanie tych operacji wymaga ingerencji projektanta w wewnętrzną reprezentację projektowanego obiektu.

W przedstawionym tutaj podejściu interakcja projektanta z systemem wspomagania procesu projektowego zachodzi na poziomie reprezentacji diagramowej. Dlatego graf funkcyjny pełni jedynie rolę pomocniczą. Projektant tworzy strukturę diagramu stosując kolejne akcje projektowe i biorąc pod uwagę wymagania zawarte w grafie funkcyjnym.

Przykład 3.1: rozkłady pomieszczeń

Graf funkcyjny reprezentujący podstawowe funkcje, jakie powinien spełniać rozkład pomieszczeń projektowanego domu (część komunikacyjna, sypialna, kuchenna, rekreacyjna oraz garaż) jest przedstawiony na rys. 3.1. Diagram uzyskany na jego podstawie w jednym

z pierwszych kroków projektowych jest pokazany na rys. 2.6b. Wielokąt etykietowany przez *Sa* reprezentuje obszar sypialny, natomiast wielokąt o etykiecie *La* reprezentuje obszar, który będzie pełnił jednocześnie funkcję rekreacyjną, komunikacyjną i kuchenną. ◊



Rys. 3.1. Graf funkcyjny rozkładu pomieszczeń
Fig. 3.1. A functional graph of a floor layout

Wykorzystanie edytora diagramowego zarówno do generowania diagramów odpowiadających projektowanym obiektom, jak i do ich modyfikacji wymaga kompatybilności pomiędzy językiem wizualnym a jego wewnętrzną reprezentacją. Projektowanie zwykle odbywa się metodą top-down, która odzwierciedla hierarchiczną naturę tego procesu. W pierwszym etapie strukturę złożonych obiektów przedstawia się na wyższym poziomie szczegółowości, a dopiero w kolejnych krokach struktura ta jest doprecyzowywana. Podobnie, definiowanie poszczególnych elementów obiektu, które będą spełniały wymagania przedstawione wstępnie w grafie funkcyjnym, odbywa się na kolejnych etapach projektowania. Z tego powodu wygodną reprezentację wewnętrzną projektowanych obiektów stanowią hierarchiczne hipergrafy [Ślus03, GLŁŚ07].

3.1. Atrybutowane hierarchiczne hipergrafy rozmieszczenia

W niniejszej pracy wewnętrzną reprezentację struktur projektów odpowiadających diagramom będą stanowiły atrybutowane hierarchiczne hipergrafy rozmieszczenia. Ten rodzaj grafów pozwala wyrażać wieloargumentowe relacje pomiędzy obiektami aktywnymi będącymi fragmentami obiektów odpowiadających różnym częściom składowym diagramów. Zawarte w hipergrafach informacje dotyczące hierarchicznych zależności między obiektami umożliwiają przedstawianie diagramów na różnych poziomach szczegółowości. Wykorzystywany tu formalny model hierarchicznych hipergrafów został wprowadzony w [DrHP00], a następnie rozszerzony w [Ślus03] w celu wyrażania relacji wieloargumentowych pomiędzy elementami znajdującymi się na różnych poziomach hierarchii.

Hipergrafy wykorzystywane w proponowanym systemie projektowym posiadają dwa rodzaje etykietowanych hiperkrawędzi. Hiperkrawędzie pierwszego rodzaju, nazywane hiperkrawędziami obiektowymi, odpowiadają obiektom projektowym reprezentowanym przez komponenty diagramu. Hiperkrawędzie drugiego rodzaju, nazywane hiperkrawędziami relacyjnymi, reprezentują relacje pomiędzy obiektami aktywnymi będącymi fragmentami obiek-

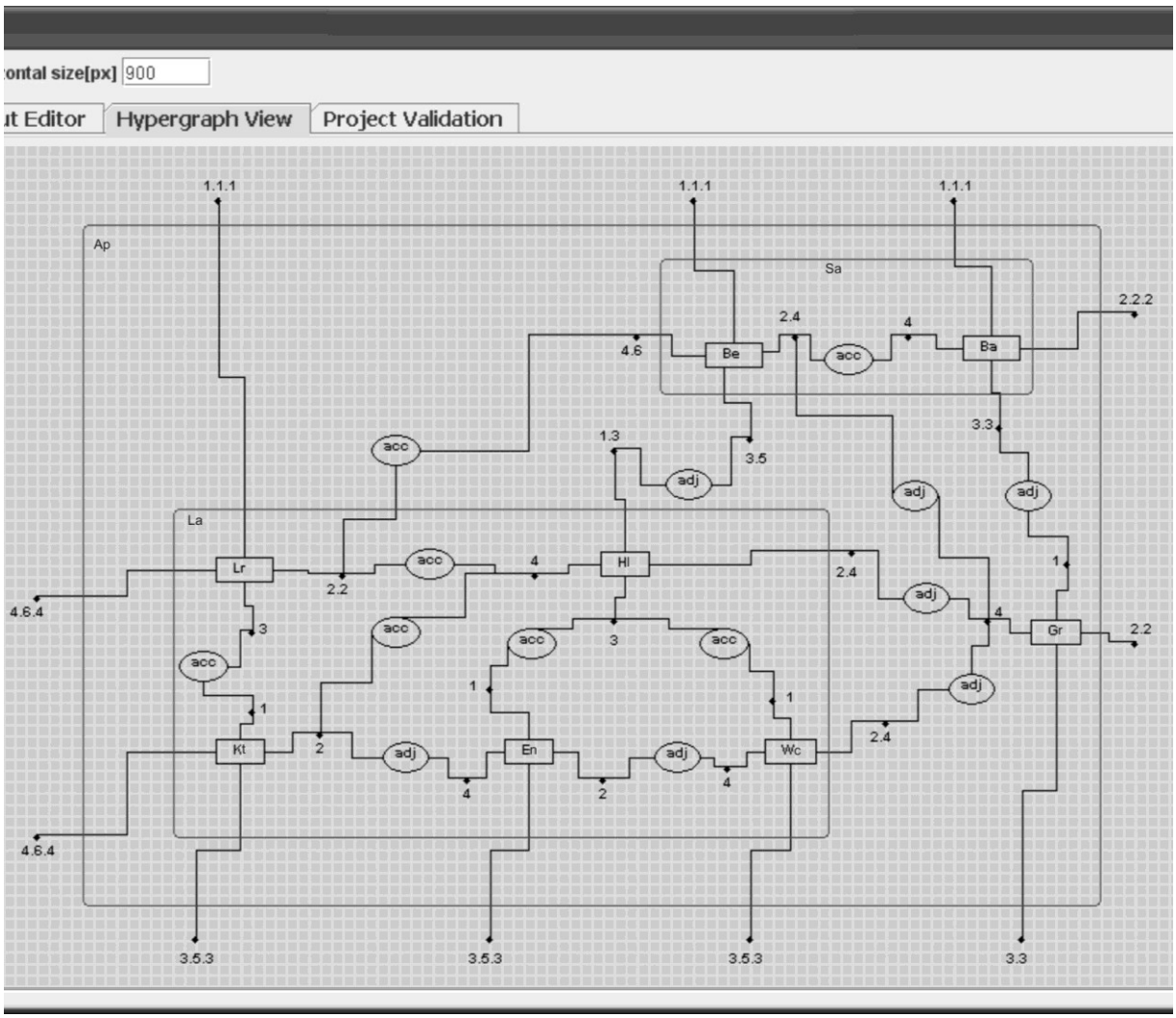
tów projektowych reprezentowanych przez hiperkrawędzie obiektowe. Wierzchołki hipergrafu odpowiadają obiektom aktywnym, które mogą być argumentami relacji. Hiperkrawędzie są etykietowane obiektami projektowymi, którym odpowiadają komponenty diagramu, i relacjami zdefiniowanymi w konceptualizacji projektowej, natomiast wierzchołki są etykietowane za pomocą obiektów aktywnych. Każdej hiperkrawędzi jest przypisany różnowartościowy ciąg wierzchołków docelowych. Wierzchołki docelowe hiperkrawędzi obiektowej reprezentują obiekty aktywne przypisane obiektowi reprezentowanemu przez tę hiperkrawędź. Hiperkrawędzie relacyjne mogą być skierowane lub nieskierowane w zależności od tego, czy reprezentowane przez nie relacje są symetryczne. Skierowanej hiperkrawędzi relacyjnej, oprócz ciągu wierzchołków docelowych, jest przypisany także różnowartościowy ciąg wierzchołków źródłowych. W przypadku projektowania rozkładów pomieszczeń, krzeseł i wież przesyłowych rozważane tu relacje są z natury symetryczne, więc są one reprezentowane przez hiperkrawędzie nieskierowane. Ponieważ na wyższym poziomie szczegółowości hipergraf może być traktowany jako hiperkrawędź, dla każdego hierarchicznego hipergrafu jest wyznaczony ciąg wierzchołków zewnętrznych, którego długość określa typ tego hipergrafu. Hiperkrawędzie obiektowe mogą zawierać zagnieżdżone w nich hipergrafy. Takie hierarchiczne krawędzie o niepustej zawartości reprezentują złożone obiekty, którym odpowiadają komponenty pełniące określone funkcje lub grupy komponentów. Zagnieżdżony hipergraf reprezentuje wewnętrzną strukturę obiektu odpowiadającego hierarchicznej hiperkrawędzi.

W celu reprezentacji charakterystycznych cech obiektów są wykorzystywane atrybuty przypisywane hiperkrawędziom obiektowym i wierzchołkom. Reprezentują one takie własności obiektów, jak na przykład kształt, rozmiar, pozycja, liczba okien i drzwi, kolor, materiał, tekstura. Wartości przypisane poszczególnym atrybutom pozwalają określić rozmieszczenie komponentów diagramu w przestrzeni i dzięki temu odpowiednio odwzorowywać hipergrafy w różne modele graficzne projektowanych artefaktów.

Przykład 3.2: rozkłady pomieszczeń

Wewnętrzna reprezentacja struktury projektu odpowiadającego diagramowi przedstawionemu na rys. 2.1b jest pokazana na rys. 3.2. Widoczny tu hipergraf składa się z jedenastu hiperkrawędzi reprezentujących obiekty odpowiadające komponentom diagramu (przedstawionych jako prostokąty), z których trzy są hierarchiczne, i czternastu hiperkrawędzi reprezentujących relacje (przedstawionych jako elipsy), z których połowa reprezentuje relację dostępności, a druga połowa relację przylegania. Hierarchiczne hiperkrawędzie reprezentują obiekty projektowe typu *zone* (etykietowane *Ap*, *La*, *Sa)*, które były dodawane do struktury projektu w trakcie projektowania rozkładu pomieszczeń metodą top-down. Hiperkrawędzie, które nie są hierarchiczne, reprezentują obiekty typu *room.* Wierzchołki, które nie są przypisane hiperkrawędziom relacyjnym, stanowią wierzchołki zewnętrzne tego hipergrafu. Hiperkrawędzie

obiektowe i wierzchołki są etykietowane elementami ze zbioru *O* konceptualizacji dla języka służącego projektowaniu rozkładów pomieszczeń, a hiperkrawędzie relacyjne są etykietowane jako *acc* lub *adj*. Nie występują tu hiperkrawędzie relacyjne reprezentujące relację *cnts* pomiędzy obiektami typu *zone* lub *room*, ponieważ relacja ta jest wyrażona poprzez hierarchię hipergrafu. Hiperkrawędź relacyjna łącząca wierzchołek 2.2 hiperkrawędzi etykietowanej *Lr* i wierzchołek 4.6 hiperkrawędzi etykietowanej *Be* jest jedną z hiperkrawędzi wyrażających relacje pomiędzy obiektami reprezentowanymi przez hiperkrawędzie zagnieżdżone w różnych hiperkrawędziach nadrzędnych.



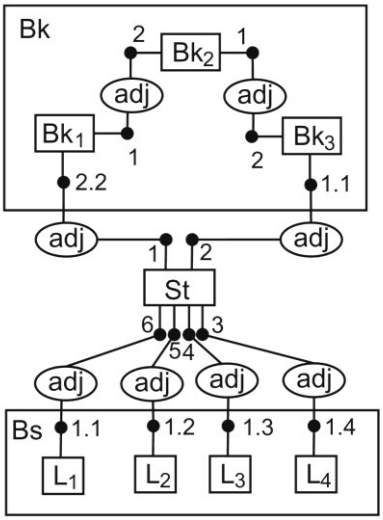
Rys. 3.2. Hierarchiczny hipergraf rozmieszczenia (w programie *HSSDR*) reprezentujący strukturę projektu odpowiadającą diagramowi z rys. 2.1b
Fig. 3.2. A hierarchical layout hypergraph (in *HSSDR* program) representing the project structure corresponding to the diagram from fig. 2.1b

Hiperkrawędziom obiektowym są przypisane atrybuty *area* i *position* określające powierzchnię i lokalizację pomieszczeń. Wierzchołki hipergrafu są atrybutowane przedstawionymi na rysunku ciągami cyfr określającymi porządek ścian i kolejne ich podziały w trakcie

projektowania (atrybut *order*). Pierwszy element tego ciągu oznacza numer ściany odpowiadającej jednemu z boków wielokąta, a długość ciągu określa, na jakim etapie projektowania ściana się pojawiła. Ponadto, wierzchołkom są przypisane także atrybuty *length*, *position*, *orientation*, *window_number*, *door_number* określające długość, lokalizację i orientację ścian odpowiadających wierzchołkom oraz liczbę znajdujących się na nich okien i drzwi. ◊

Przykład 3.3: krzesła

Hipergraf będący wewnętrzną reprezentacją struktury projektu odpowiadającej diagramowi z rys. 2.2b jest pokazany na rys. 3.3. Zawiera on dwie hiperkrawędzie hierarchiczne, z których jedna (etykietowana *Bk*) reprezentuje oparcie krzesła i zawiera trzy hiperkrawędzie reprezentujące trzy części oparcia, a druga (etykietowana *Bs*) reprezentuje podstawę i zawiera cztery hiperkrawędzie reprezentujące nogi krzesła. Hiperkrawędź etykietowana *St* reprezentuje siedzisko. Osiem hiperkrawędzi relacyjnych reprezentuje przyleganie poszczególnych części krzesła. Wierzchołki hipergrafu przypisane hiperkrawędziom odpowiadającym elementom krzesła reprezentują punkty połączenia odpowiadające fragmentom brył, poprzez które bryły te będą ze sobą łączone. Hiperkrawędziom obiektowym są przypisane atrybuty *width*, *height*, *depth* określające rozmiary komponentów oraz *colour*, *material*, *texture,* których wartości specyfikują wygląd obiektów. Wierzchołkom jest przypisany atrybut *position*, określający lokalizację punktów połączenia brył oraz atrybut *order* (widoczny na rysunku) specyfikujący kolejność punktów połączeń. ◊

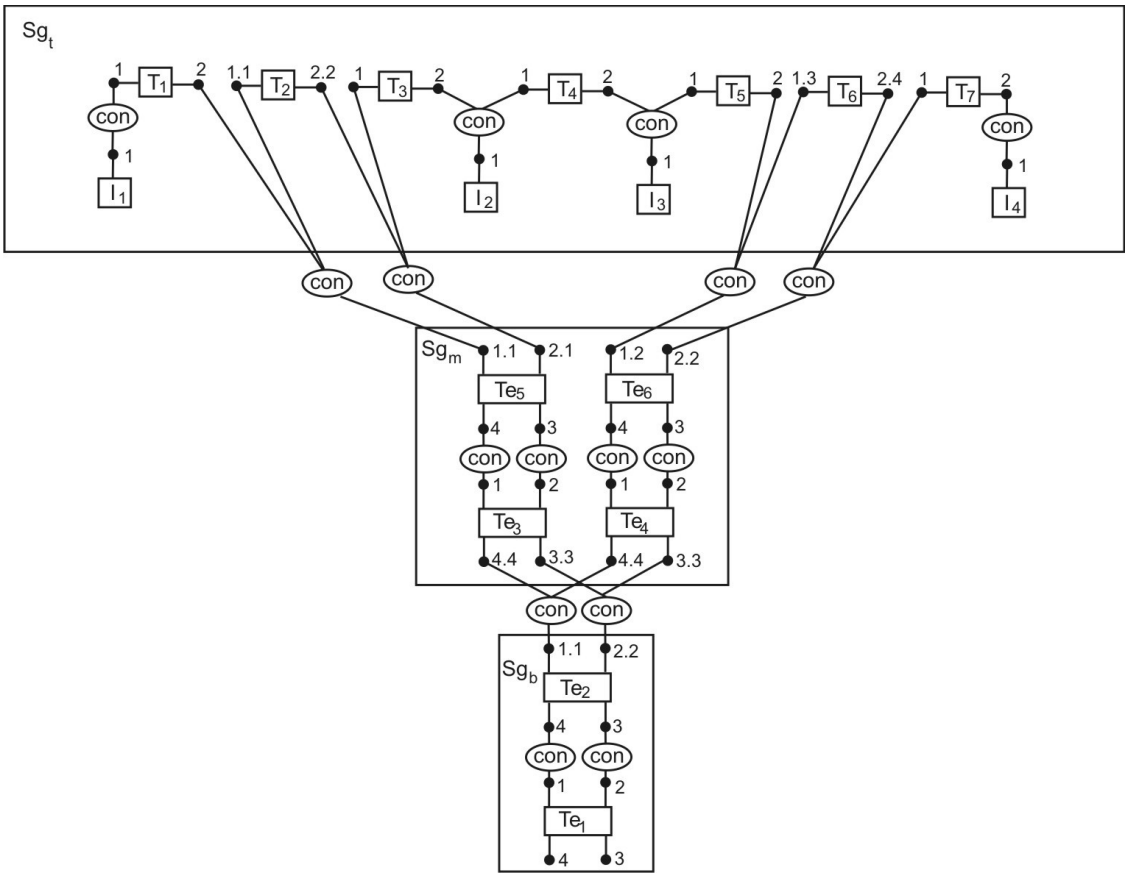


Rys. 3.3. Hierarchiczny hipergraf reprezentujący strukturę projektu odpowiadającą diagramowi z rys. 2.2b

Fig. 3.3. A hierarchical hypergraph representing the project structure corresponding to the diagram from fig. 2.2b

Przykład 3.4: wieże przesyłowe

Hipergraf będący wewnętrzną reprezentacją struktury projektu odpowiadającej diagramowi z rys. 2.3b jest pokazany na rys. 3.4. Zawiera on trzy hiperkrawędzie hierarchiczne etykietowane Sg_b, Sg_m, Sg_t i reprezentujące odpowiednio trzy segmenty wieży. Hiperkrawędź Sg_b zawiera dwie hiperkrawędzie reprezentujące elementy skratowania, a hiperkrawędź Sg_m zawiera cztery hiperkrawędzie reprezentujące elementy skratowania. Hiperkrawędź Sg_t zawiera siedem hiperkrawędzi reprezentujących skratowania poziome i cztery hiperkrawędzie reprezentujące izolatory. Wierzchołki hipergrafu reprezentują węzły, poprzez które skratowania i izolatory są ze sobą łączone. Szesnaście hiperkrawędzi relacyjnych reprezentuje zespolenie poszczególnych węzłów skratowań. Hiperkrawędziom obiektowym są przypisane atrybuty *width*, *height*, *position*, *type*, *material* określające rozmiary, lokalizację i wygląd obiektów. Wierzchołkom jest przypisany atrybut *order* specyfikujący kolejność węzłów względem obiektów, do których są przypisane. ◊



Rys. 3.4. Hierarchiczny hipergraf reprezentujący strukturę projektu odpowiadającą wieży przesyłowej z rys. 2.3b

Fig. 3.4. A hierarchical hypergraph representing the project structure corresponding to the transmission tower from fig. 2.3b

Przedstawiona poniżej formalna definicja atrybutowanego hierarchicznego hipergrafu rozmieszczenia jest modyfikacją definicji występujących uprzednio w pracach [GLŁŚ07, GrŚL08].

Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową. Niech $\Sigma = O \cup R$ będzie ustalonym alfabetem etykiet odpowiednio dla elementów hipergrafu reprezentujących obiekty i relacje. Niech $[i]$ oznacza zbiór $\{1,2,..., i\}$ dla $i \geq 0$ (gdzie $[0] = \varnothing$).

Definicja 3.1:

Atrybutowany hierarchiczny hipergraf rozmieszczenia nad Σ i A jest systemem $G = (E, V, t, s, lb, att, ext, ch)$, gdzie:

1. $E = E_C \cup E_R$, gdzie $E_C \cap E_R = \varnothing$, jest niepustym, skończonym zbiorem hiperkrawędzi, gdzie elementy z E_C (hiperkrawędzie obiektowe) reprezentują obiekty odpowiadające komponentom diagramu, a elementy z E_R (hiperkrawędzie relacyjne) reprezentują relacje,
2. V jest niepustym, skończonym zbiorem wierzchołków reprezentujących obiekty aktywne,
3. $t: E \to V^*$ jest odwzorowaniem przypisującym hiperkrawędziom różnowartościowe ciągi wierzchołków docelowych,
4. $s: E_R \to V^*$ jest odwzorowaniem przypisującym hiperkrawędziom relacyjnym różnowartościowe ciągi wierzchołków źródłowych,
5. $lb = lb_O \cup lb_R$, gdzie:
   - $lb_O: E_C \cup V \to O$ jest funkcją etykietowania hiperkrawędzi obiektowych i wierzchołków,
   - $lb_R: E_R \to R$ jest funkcją etykietowania hiperkrawędzi relacyjnych,
6. $att: E_C \cup V \to 2^A$ jest funkcją atrybutowania hiperkrawędzi obiektowych i wierzchołków zgodną z funkcją Att z konceptualizacji $\mathcal{C}$, tzn. $\forall\, at \in E_C \cup V$, gdzie $lb_O(at) = o$, $att(at) = Att(o)$,
7. $ext: [n] \to V^*$ jest odwzorowaniem wyznaczającym ciąg zewnętrznych wierzchołków hipergrafu,
8. $ch: E_C \to 2^{AT}$ jest funkcją zagnieżdżenia, gdzie $AT = E \cup V$ jest zbiorem atomów hipergrafu, taką że następujące warunki są spełnione:
   - $\forall\, at \in AT\ \forall\, e_1, e_2 \in E_C\ \ at \in ch(e_1) \wedge at \in ch(e_2) \Rightarrow e_1 = e_2$, tzn. żaden atom nie może być zagnieżdżony w dwóch różnych hiperkrawędziach,
   - $\forall\, e \in E_C\ \ e \notin ch^+(e)$, gdzie $ch^+(e)$ oznacza wszystkich potomków hiperkrawędzi e, tzn. hiperkrawędź nie może być swoim własnym potomkiem,
   - $\forall\, e_1, e_2 \in E_C: e_1 \in ch(e_2) \Rightarrow$ wszystkie wierzchołki występujące w $t(e_1)$ należą do $ch(e_2)$, tzn. wierzchołki docelowe zagnieżdżonej hiperkrawędzi obiektowej e_1 muszą być zagnieżdżone w tej samej hiperkrawędzi co e_1. □

Obiekty projektowe odpowiadające komponentom rozwiązania wraz z określonymi dla nich obiektami aktywnymi są reprezentowane przez uchwyty hipergrafu rozmieszczenia, które są definiowane jako hiperkrawędzie obiektowe wraz ze wszystkimi przypisanymi im wierzchołkami oraz z atomami potomnymi. Dla każdego uchwytu jest wyznaczona sekwencja wierzchołków zewnętrznych. Natomiast relacje pomiędzy obiektami aktywnymi określone w strukturze projektu są reprezentowane przez hiperkrawędzie relacyjne hipergrafu łączące wierzchołki zewnętrzne należące do odpowiednich uchwytów hipergrafu.

Niech e będzie hiperkrawędzią obiektową hipergrafu rozmieszczenia G i niech $T(e)$ oznacza zbiór wierzchołków występujących w sekwencji $t(e)$ przypisanej tej hiperkrawędzi funkcją t.

Definicja 3.2:

Uchwyt indukowany przez hiperkrawędź obiektową e jest atrybutowanym hierarchicznym hipergrafem rozmieszczenia nad Σ i A postaci:

$G(e) = (E, V, t, s, lb, att, ext, ch)$, gdzie $e \in E_C$ i $E \cup V = \{e\} \cup ch^+(e) \cup T(e)$. □

Hierarchiczny hipergraf rozmieszczenia, w którym są określone wartości wszystkich atrybutów przypisanych jego hiperkrawędziom i wierzchołkom, jest nazywany *wartościowanym* hipergrafem rozmieszczenia.

Definicja 3.3:

Niech $G = (E, V, t, s, lb, att, ext, ch)$ będzie atrybutowanym hierarchicznym hipergrafem rozmieszczenia nad Σ i A. Niech Ev^G będzie rodziną funkcji częściowych przypisujących wartości atrybutów hiperkrawędziom obiektowym i wierzchołkom hipergrafu G, taką że $\forall\ a \in A\ \ Ev^G_a : (E_C \cup V) \times A \to D_a$, gdzie D_a jest zbiorem wartości atrybutu a.

Para (G, Ev^G) jest nazywana **wartościowanym hierarchicznym hipergrafem rozmieszczenia.** □

Podczas gdy hierarchiczne hipergrafy rozmieszczenia reprezentujące struktury projektów odpowiadają syntaktyce języka wizualnego, wartościowane hierarchiczne hipergrafy rozmieszczenia reprezentują diagramy będące elementami języka wizualnego. Wartościowany hierarchiczny hipergraf rozmieszczenia (G, Ev^G) reprezentuje diagram Dg z funkcją Ev' wartościowania atrybutów jego komponentów, jeśli $\forall\ at \in E_C \cup V \quad \forall\ a \in att(at)$ $Ev^G_a(at, a) = Ev'_a(f(s), a)$, gdzie $lb_O(at) = o$ i $\tau^1_{Str}(o, Att_Q(o)) = (s, f)$), tzn. wartości wszystkich atrybutów każdego atomu hipergrafu etykietowanego nazwą obiektu o są takie same jak wartości tych atrybutów dla komponentu diagramu wyznaczonego dla obiektu o przez realizację τ^1_{Str} struktury projektu Str, do której należy o. Modyfikacje diagramów dokonywane w trakcie procesu projektowego powodują zarówno zmiany struktury odpowiadających im

hipergrafów, jak i zmiany wartości atrybutów przypisanych atomom (hiperkrawędziom obiektowym i wierzchołkom) hipergrafów.

3.2. Realizacja hierarchicznych hipergrafów rozmieszczenia

W rozdziale 7 zostanie opisany system wspomagania projektowania wyposażony w agentów będących inteligentnymi asystentami projektanta, którzy działają na wewnętrznej reprezentacji projektów w postaci hierarchicznych hipergrafów rozmieszczenia. Automatycznie generują oni zbiory hipergrafów odpowiadających proponowanym rozwiązaniom. Jednak aby projektant mógł dokonać oceny tych rozwiązań, wygenerowane hipergrafy muszą zostać zwizualizowane.

Dla hierarchicznych hipergrafów rozmieszczenia jest definiowana realizacja wyznaczająca możliwe sposoby ich odwzorowania w modele graficzne [Ślus99, GrŚP03]. Określa ona semantyczne znaczenie atomów hipergrafu poprzez przypisanie hiperkrawędziom reprezentującym obiekty projektowe prymitywów geometrycznych oraz wyznaczenie odpowiedniości pomiędzy hiperkrawędziami relacyjnymi i relacjami przestrzennymi zachodzącymi między odpowiednimi prymitywami. Przypisanie to musi być zgodne z wartościami nadanymi atrybutom atomów hipergrafu. Realizacja umożliwia przestrzenne rozmieszczenie i dopasowanie do siebie komponentów projektowanego obiektu w sposób zgodny z przyjętymi kryteriami projektowymi.

Niech $DG = (S, F, S', A', Att', X)$ będzie dziedziną diagramów projektowych.

Definicja 3.4:
Niech $G = (E, V, t, s, lb, att, ext, ch)$ będzie atrybutowanym hierarchicznym hipergrafem rozmieszczenia nad Σ i A.

Realizacja dla G jest funkcją $I = (I_O, I_R)$, gdzie:

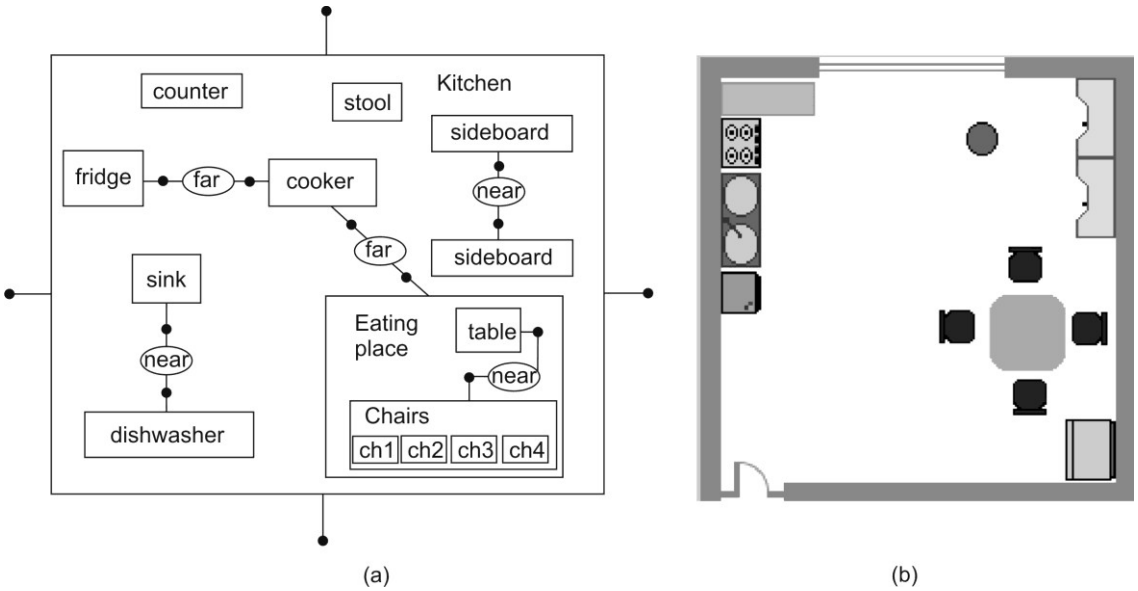
1. I_O: $(E_C \cup V) \times 2^A \rightarrow S \times F$, przypisuje prymitywy geometryczne wraz z określoną dla nich transformacją atrybutowanym hiperkrawędziom obiektowym i wierzchołkom reprezentującym komponenty diagramu,
2. I_R: $E_R \rightarrow X$ przypisuje relacje przestrzenne hiperkrawędziom relacyjnym. □

Przykład 3.5: aranżacja pomieszczeń
Hierarchiczny hipergraf rozmieszczenia reprezentujący ustawienie mebli i sprzętów w kuchni jest przedstawiony na rys. 3.5a. Zawiera on trzy hierarchiczne hiperkrawędzie. Hierarchiczna hiperkrawędź etykietowana *Kitchen* reprezentuje pomieszczenie, na powierzchni którego sprzęty powinny być rozmieszczone. Hiperkrawędź etykietowana *Eating place* zawiera hierarchiczny hipergraf złożony z hiperkrawędzi reprezentującej stół, która jest połączona hiper-

krawędzią relacyjną z hierarchiczną hiperkrawędzią etykietowaną *Chairs* reprezentującą grupę czterech krzeseł. Pozostałe hiperkrawędzie reprezentują pojedyncze sprzęty lub relacje.

Hiperkrawędzie relacyjne reprezentują odległość pomiędzy sprzętami. Są one etykietowane *near* lub *far*, co wskazuje, które sprzęty powinny być umieszczone blisko siebie, a które możliwie daleko od siebie. Na przykład, szafki powinny być umieszczone blisko siebie, natomiast kuchenka i lodówka z daleka od siebie. Należy zauważyć, że jeśli hiperkrawędź relacyjna jest incydentna z wierzchołkiem połączonym z hierarchiczną hiperkrawędzią, wówczas relacja przez nią reprezentowana jest dziedziczona przez wszystkie hiperkrawędzie znajdujące się w hiperkrawędzi hierarchicznej (np. wszystkie meble w kąciku jadalnym powinny być umieszczone z dala od kuchenki).

Kształty, kolory oraz współrzędne i sposób lokalizacji mebli i sprzętów są określone odpowiednio przez wartości atrybutów *shape*, *colour*, *position* i *location*, przypisanych wszystkim niehierarchicznym hiperkrawędziom reprezentującym obiekty. Atrybut *location* może przyjmować wartości 0, 1 lub 2, co odpowiednio oznacza, że nie ma ograniczeń co do umiejscowienia danego elementu (np. taborety, krzesła), element powinien być umieszczony pod ścianą (np. kuchenka, lodówka, zlew), element może być umieszczony pod ścianą lub pod oknem (np. stół, lada).



Rys. 3.5. Aranżacja kuchni: a) hierarchiczny hipergraf reprezentujący rozmieszczenie sprzętów w kuchni, b) wizualizacja tego hipergrafu

Fig. 3.5. A kitchen arrangenent: a) a hierarchical hypergraph representing layout of kitchen pieces, b) a visualization of this hypergraph

Realizacja zdefiniowana dla tego hipergrafu przypisuje hiperkrawędziom reprezentującym obiekty ikony odpowiadające sprzętom i meblom zgodnie z etykietami tych hiperkrawędzi. Zbiór ikon reprezentujących lodówkę, kuchenkę, zlew, taboret, stół, ladę, szafkę i zmywarkę jest pokazany na rys. 3.6. Hiperkrawędziom reprezentującym obiekty są przypisane

także transformacje, będące złożeniem przesunięcia i obrotu i pozwalające rozmieścić ikony zgodnie z relacjami określonymi przez hiperkrawędzie relacyjne i wartościami atrybutu *location* przypisanego tym hiperkrawędziom. Jedna z możliwych wizualizacji hipergrafu w postaci modelu graficznego otrzymana z wykorzystaniem opisanej realizacji jest przedstawiona na rys. 3.5b. ◊

Rys. 3.6. Ikony reprezentujące sprzęty kuchenne
Fig. 3.6. Icons representing kitchen pieces

Przykład 3.6: krzesła
Realizacja zdefiniowana dla hierarchicznego hipergrafu z rys. 3.3 przypisuje hiperkrawędziom reprezentującym elementy krzesła prymitywy geometryczne ze zbioru zawierającego prostopadłościan, walec, stożek i kulę. Atrybuty wierzchołków przypisanych tym hiperkrawędziom określają punkty, w których odpowiadające im prymitywy powinny być ze sobą połączone. Hiperkrawędziom reprezentującym poszczególne części krzesła są przypisane także transformacje, będące złożeniem niejednorodnego skalowania, przesunięcia i obrotu. Skalowanie pozwala uzyskać odpowiedni kształt elementów krzesła, a przesunięcie i obrót pozwalają rozmieścić bryły, zgodnie z relacją przylegania określoną przez hiperkrawędzie relacyjne i wartościami atrybutu *position* przypisanego wierzchołkom tych hiperkrawędzi. Kilka możliwych modeli krzeseł, będących wizualizacjami tego samego hipergrafu (rys. 3.3) otrzymanych z wykorzystaniem opisanej realizacji, jest przedstawionych na rys. 3.7. Różny wygląd modeli został spowodowany zmianą brył przypisanych hiperkrawędziom reprezentującym elementy krzeseł, zmianą wartości atrybutów przypisanych tym hiperkrawędziom i określających kolor, materiał oraz teksturę elementów krzeseł, a także różnymi transformacjami przypisanymi hiperkrawędziom za pomocą realizacji. ◊

Przykład 3.7: wieże przesyłowe
Realizacja zdefiniowana dla hierarchicznego hipergrafu z rys. 3.4 przypisuje hiperkrawędziom reprezentującym elementy skratowania, skratowania i izolatory kształty odpowiadające typom tych elementów określonym w atrybutach hiperkrawędzi. Atrybuty wierzchołków przypisanych tym hiperkrawędziom określają węzły, w których odpowiadające im kształty powinny być ze sobą połączone. Hiperkrawędziom reprezentującym poszczególne części wieży są przypisane także transformacje pozwalające uzyskać odpowiedni kształt komponentów wieży i ich rozmieszczenie zgodne z relacją zespolenia określoną przez hiperkrawędzie relacyjne. Jedna z możliwych wizualizacji hipergrafu z rys. 3.4 w postaci diagramu projektowego otrzymana z wykorzystaniem opisanej realizacji jest przedstawiona na rys. 2.3b. ◊

Rys. 3.7. Modele krzeseł odpowiadające temu samemu hierarchicznemu hipergrafowi
Fig. 3.7. Chair models corresponding to the same hierarchical hypergraph

3.3. Podsumowanie

W rozdziale tym zostały zdefiniowane i omówione atrybutowane hierarchiczne hipergrafy rozmieszczenia stanowiące wewnętrzną reprezentację struktur projektów odpowiadających diagramom. Hipergrafy te pozwalają reprezentować relacje wieloargumentowe pomiędzy obiektami projektowymi, natomiast atrybuty przypisane hiperkrawędziom i wierzchołkom hipergrafów reprezentują własności tych obiektów. Zostało przedstawione pojęcie uchwytu hipergrafu rozmieszczenia indukowanego przez jego hiperkrawędź obiektową. Uchwyt taki reprezentuje obiekt projektowy odpowiadający komponentowi diagramu wraz z określonymi dla niego obiektami aktywnymi, a jego struktura jest wykorzystywana podczas stosowania produkcji hierarchicznych gramatyk rozmieszczenia, które zostaną omówione w rozdziale 5.

Zostało także wprowadzone pojęcie wartościowanego hierarchicznego hipergrafu rozmieszczenia, którego atrybuty posiadają konkretne wartości określające semantykę diagramów. Taka reprezentacja wewnętrzna diagramów jest łatwa do automatycznego przetwarzania, a jednocześnie pozwala przechowywać zarówno syntaktyczną, jak i semantyczną wiedzę projektową. Zostały zamieszczone przykłady hipergrafów reprezentujących diagramy projektowe omawiane w rozdziale 2. Została również zdefiniowana realizacja hierarchicznych hipergrafów rozmieszczenia wyznaczająca sposób odwzorowania tych hipergrafów w modele graficzne.

Przedstawiona w tym rozdziale reprezentacja wewnętrzna diagramów projektowych umożliwia stworzenie struktury relacyjnej, której elementy są przypisywane składowym formuł logicznych opisujących diagramy. Formuły te są wykorzystywane w logicznym modelu wnioskowania pozwalającym systemowi wspomagającemu projektowanie decydować o poprawności projektów. Struktura relacyjna i formuły logiczne wyrażające wiedzę o diagramach zostaną omówione w rozdziale 6.

4. OPERACJE NA HIERARCHICZNYCH HIPERGRAFACH

W trakcie procesu projektowego projektant często dostrzega nowe możliwości i zmienia cele projektowe oraz odpowiednio modyfikuje diagram projektowy, zanim uzyska satysfakcjonujące go rozwiązanie. Aby odzwierciedlić zmiany projektu związane z dynamiką procesu projektowego w wewnętrznej reprezentacji wizualnego języka projektowania, należy zdefiniować operacje działające na hierarchicznych hipergrafach rozmieszczenia. Operacje te pozwalają tworzyć i modyfikować hipergrafy reprezentujące struktury projektowanych obiektów. W rozdziale tym zostaną przedstawione operacje na hipergrafach odpowiadające podziałowi obiektu na części składowe i scalaniu kilku obiektów w jeden obiekt [GLŁŚ07, GrŚL08], a także zostaną zdefiniowane dwie nowe operacje odpowiadające odpowiednio dodawaniu obiektów do struktury diagramu i ich usuwaniu.

Pierwszy narysowany przez użytkownika diagram projektowy jest automatycznie przekształcany na hipergrafową reprezentację struktury projektu. Wszystkim kolejnym modyfikacjom diagramu, będącym odzwierciedleniem nowych idei projektowych [DoGr01] i dokonywanym przez użytkownika w trakcie procesu projektowania, będą odpowiadały zmiany w hipergrafowej strukturze danych. Dla zmodyfikowanego diagramu reprezentacja hipergrafowa nie jest tworzona od nowa, lecz powstaje poprzez wykonanie pewnych operacji na hipergrafie odpowiadającym poprzedniemu diagramowi.

4.1. Operacja rozwinięcia hiperkrawędzi hipergrafu

Podstawową operacją wykonywaną na hipergrafie podczas procesu projektowego jest *rozwinięcie hiperkrawędzi*, które zostało zdefiniowane w [GLŁŚ07]. Operacja ta odpowiada podziałowi obiektu odpowiadającemu pewnemu komponentowi diagramu na inne obiekty i pozwala reprezentować strukturę projektu w sposób bardziej szczegółowy. Zagnieżdża ona hierarchiczny hipergraf reprezentujący składowe części obiektu i relacje zachodzące między nimi w hiperkrawędzi danego hipergrafu reprezentującej dzielony obiekt. Wierzchołki połączone z rozwijaną hiperkrawędzią są zastępowane odpowiednimi wierzchołkami zewnętrznymi zagnieżdżanego hipergrafu. Atrybut *order* wierzchołków zagnieżdżonego hipergrafu,

które zastępują wierzchołki rozwijanej hiperkrawędzi, zostaje uzupełniony o wartość tego atrybutu, jaką posiadały zastępowane wierzchołki.

Niech dla danego atrybutowanego hierarchicznego hipergrafu rozmieszczenia $G = (E, V, t, s, lb, att, ext, ch)$, $S(e)$ i $T(e)$ oznaczają odpowiednio zbiory wierzchołków źródłowych i docelowych hiperkrawędzi e występujące w sekwencjach $s(e)$ i $t(e)$.

Definicja 4.1:

Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ i $H = (E_H, V_H, t_H, s_H, lb_H, att_H, ext_H, ch_H)$ będą dwoma hierarchicznymi hipergrafami rozmieszczenia nad Σ i A.

Niech $order \in A$ będzie atrybutem wierzchołkowym ze zbiorem wartości D_{order}. Niech $\bar{e} \in E_G^C$ będzie hiperkrawędzią obiektową z G taką, że $ch_G(\bar{e}) = \emptyset$ i niech EXT_H oznacza zbiór wierzchołków zewnętrznych H.

Operacja rozwinięcia hiperkrawędzi jest zdefiniowana za pomocą trzech funkcji:

1. funkcja rozwinięcia dev: $EXT_H \rightarrow T_G(\bar{e})$ ustala odpowiedniość między wierzchołkami docelowymi hiperkrawędzi $\bar{e}$ i wierzchołkami zewnętrznymi hipergrafu H,
2. funkcja Ev^H_{order}: $EXT_H \times order \rightarrow D_{order}$ przypisuje wartości atrybutowi $order$ wierzchołków zewnętrznych hipergrafu H w taki sposób, że $\forall\, v \in EXT_H$ $order(v) = order_H(v).order_G(dev(v))$, gdzie operator „.” oznacza konkatenację ciągów,
3. funkcja osadzenia rozwinięcia emb_{dev}: $E_1 \rightarrow 2^{E_2}$, gdzie:
   - $E_1 = \bigcup_{v \in dev(EXT_H)} \{e \in E_G^R \mid v \in S_G(e) \cup T_G(e)\}$,
   - E_2 jest skończonym zbiorem hiperkrawędzi relacyjnych takim, że $E_2 \cap E_G = \emptyset$,

   umożliwia zastąpienie wszystkich hiperkrawędzi relacyjnych hipergrafu G, które były połączone z wierzchołkami docelowymi hiperkrawędzi $\bar{e}$ (zbiór E_1) nowymi hiperkrawędziami relacyjnymi ze zbioru E_2, które będą połączone z wierzchołkami z EXT_H.

Wynikiem operacji rozwinięcia hiperkrawędzi jest atrybutowany hierarchiczny hipergraf rozmieszczenia $K = (E_K, V_K, t_K, s_K, lb_K, att_K, ext_K, ch_K)$ nad Σ i A, gdzie:

1. $E_K = (E_G - E_1) \cup emb_{dev}(E_1) \cup E_H$,
2. $V_K = (V_G - dev(EXT_H)) \cup V_H$,
3. t_K: $E_K \rightarrow V_K^*$ i s_K: $E_K^R \rightarrow V_K^*$ i są zdefiniowane w taki sposób, że:
   - $t_K(\bar{e}) \subseteq V^*$, gdzie $V = T_G(\bar{e}) - dev(EXT_H)$,
   - $\forall\, e \in (E_G - E_1) - \{\bar{e}\}$ $t_K(e) = t_G(e)$,
   - $\forall\, e \in (E_G - E_1) \cap E_G^R$ $s_K(e) = s_G(e)$,
   - $\forall\, e \in E_H$ $t_K(e) = t_H(e)$,
   - $\forall\, e \in E_H^R$ $s_K(e) = s_H(e)$,
   - $\forall\, e \in emb_{dev}(E_1)$ $s_K(e) = f(e) \wedge t_K(e) = g(e)$, gdzie

$$f, g\colon emb_{dev}(E_1) \to ((V_G - T_G(\bar{e})) \cup EXT_H)^*,$$

4. lb_K jest zdefiniowane następująco:
   - $\forall\ e \in (E_G - E_1)\ lb_K(e) = lb_G(e)$,
   - $\forall\ e \in emb_{dev}(E_1)\ lb_K(e) = lb_G(e_1)$, gdzie $e_1 = emb^{-1}_{dev}(e)$,
   - $\forall\ e \in E_H\ lb_K(e) = lb_H(e)$,
   - $\forall\ v \in (V_G - dev(EXT_H))\ lb_K(v) = lb_G(v)$,
   - $\forall\ v \in V_H\ lb_K(v) = lb_H(v)$,
5. att_K jest zdefiniowane następująco:
   - $\forall\ e \in E^C_G\ att_K(e) = att_G(e)$,
   - $\forall\ e \in E^C_H\ att_K(e) = att_H(e)$,
   - $\forall\ v \in (V_G - dev(EXT_H))\ att_K(v) = att_G(v)$,
   - $\forall\ v \in V_H\ att_K(v) = att_H(v)$,
6. $ext_K\colon [n_K] \to V_K^*$,
7. ch_K jest zdefiniowane następująco:
   - $ch_K(\bar{e}) = \{at \in E_H \cup V_H \mid \neg\exists\ e \in E^C_H : at \in ch_H(e)\}$,
   - $\forall\ e \in (E^C_G - \{\bar{e}\})\ ch_K(e) = ch_G(e)$,
   - $\forall\ e \in E^C_H\ ch_K(e) = ch_H(e)$. □

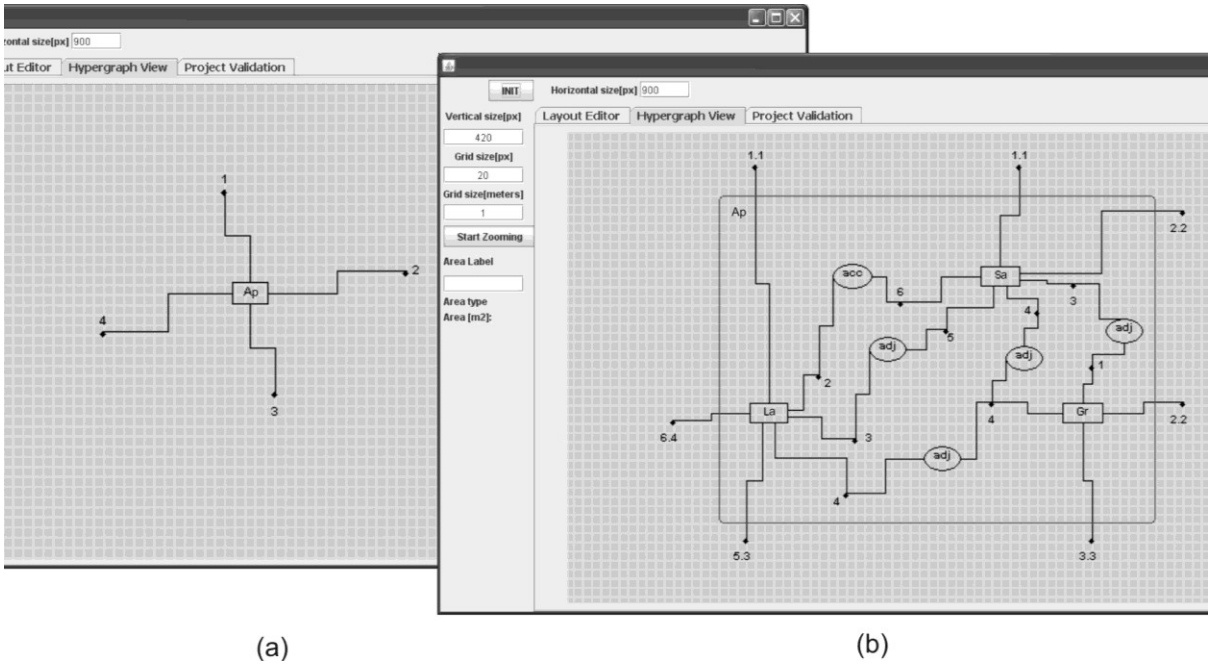
W wyniku zastosowania operacji rozwinięcia hiperkrawędzi hipergraf H zostaje zagnieżdżony w jednej z hiperkrawędzi obiektowych G, która nie była hiperkrawędzią hierarchiczną. Bezpośrednimi potomkami tej hiperkrawędzi, określonymi przez ch_K, stają się atomy hipergrafu H, które nie były dotychczas zagnieżdżone w żadnej hiperkrawędzi. Wierzchołki połączone w G z rozwijaną hiperkrawędzią są zastępowane odpowiadającymi im zewnętrznymi wierzchołkami hipergrafu H. Odpowiedniość ta jest wyznaczana przez funkcję rozwinięcia *dev* określaną na podstawie kryteriów i ograniczeń projektowych. Opisywana tu operacja może zmniejszyć liczbę wierzchołków połączonych z rozwijaną hiperkrawędzią, gdyż funkcje niektórych z nich mogą zostać przejęte przez wierzchołki połączone z zagnieżdżanymi hiperkrawędziami.

Ponadto, w wyniku omawianej operacji hiperkrawędzie reprezentujące relacje, które w sekwencji swoich wierzchołków źródłowych lub docelowych zawierały zastępowane wierzchołki rozwijanej hiperkrawędzi (wierzchołki należące do przeciwdziedziny funkcji *dev*), zostają zastąpione nowymi hiperkrawędziami reprezentującymi relacje i łączącymi pozostałe wierzchołki hipergrafu G z wierzchołkami zewnętrznymi osadzanego hipergrafu (EXT_H). Sposób zastąpienia każdej hiperkrawędzi relacyjnej odpowiadającym jej zbiorem nowych hiperkrawędzi także jest określany na podstawie wymagań projektowych.

Przykład 4.1: rozkłady pomieszczeń

Rozważmy przykład tworzenia diagramu projektowego przedstawionego na rys. 2.1b i reprezentującego rozkład pomieszczeń parterowego domu z garażem pokazany na rys. 2.1a. Pierwszy diagram rysowany przez użytkownika reprezentuje obszar całego domu. Automatycznie jest też generowany początkowy hipergraf (rys. 4.1a) reprezentujący strukturę projektu odpowiadającą temu diagramowi. Składa się on z jednej hiperkrawędzi o etykiecie *Ap* połączonej z czteroma wierzchołkami zewnętrznymi reprezentującymi boki tego obszaru i umieszczonymi na rysunku w sposób zgodny z geograficzną orientacją odpowiadających im boków w diagramie. Początkową wartość atrybutu *order* dla tych wierzchołków stanowią numery nadawane zgodnie z ruchem wskazówek zegara, zaczynając od górnego wierzchołka położonego najbardziej na lewo.

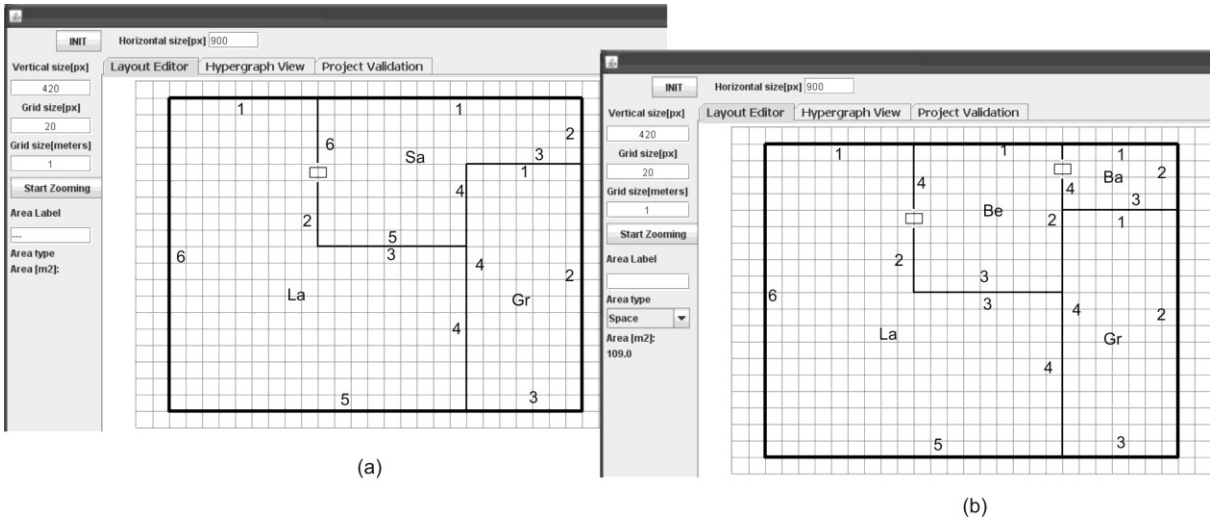
W następnym kroku projektant dzieli obszar domu na trzy obszary reprezentujące część sypialną, część rekreacyjną i garaż (rys. 4.2a). Na skutek tej akcji projektowej automatycznie jest wywoływana operacja rozwinięcia hiperkrawędzi, w wyniku której hipergraf rozmieszczenia reprezentujący wspomniane trzy obszary i relacje między nimi zostaje zagnieżdżony w hiperkrawędzi reprezentującej cały obszar. Otrzymany hierarchiczny hipergraf rozmieszczenia jest przedstawiony na rys. 4.1b.



Rys. 4.1. Hierarchiczne hipergrafy w programie *HSSDR*: a) hiperkrawędź reprezentująca cały obszar domu, b) hierarchiczny hipergraf rozmieszczenia otrzymany w wyniku zastosowania operacji rozwinięcia tej hiperkrawędzi

Fig. 4.1. Hierarchical hypergraphs in *HSSDR* program: a) a hyperedge representing the whole house area, b) a hierarchical layout hypergraph obtained as a result of applying the development operation to this hyperedge

Cztery wierzchołki zewnętrzne hiperkrawędzi przedstawionej na rys. 4.1a zostają zastąpione siedmioma zewnętrznymi wierzchołkami zagnieżdżonego hipergrafu rozmieszczenia zgodnie z geometryczną orientacją odpowiadających im ścian. Numery, będące wartościami atrybutu określającego kolejność ścian dla wierzchołków zewnętrznych hipergrafu rozmieszczenia zagnieżdżonego w hiperkrawędzi reprezentującej obszar domu, są konkatenowane z numerami oznaczającymi wartości atrybutu *order* zastąpionych przez nie wierzchołków połączonych z nadrzędną hiperkrawędzią. Wierzchołek o numerze 1 hiperkrawędzi *Ap* jest zastępowany wierzchołkami reprezentującymi północne ściany obszaru rekreacyjnego (*La*) i sypialnego (*Sa*). Oba nowe wierzchołki są numerowane 1.1, gdzie pierwsza część ciągu oznacza, że odpowiadają one pierwszym ścianom obszarów *La* i *Sa*, natomiast druga jego część jest dziedziczona po zastąpionym wierzchołku. Wierzchołek o numerze 2 jest zastępowany wierzchołkiem 2 obszaru sypialnego i wierzchołkiem 2 garażu. Oba te wierzchołki reprezentują wschodnie boki diagramu i są numerowane 2.2. Wierzchołek o numerze 3 jest zastępowany wierzchołkiem 3 garażu (numerowanym 3.3) i wierzchołkiem 5 obszaru rekreacyjnego (numerowanym 5.3). Wierzchołek o numerze 4 jest zastępowany wierzchołkiem o numerze 6 obszaru rekreacyjnego (numerowanym 6.4), który reprezentuje teraz zachodni bok diagramu.



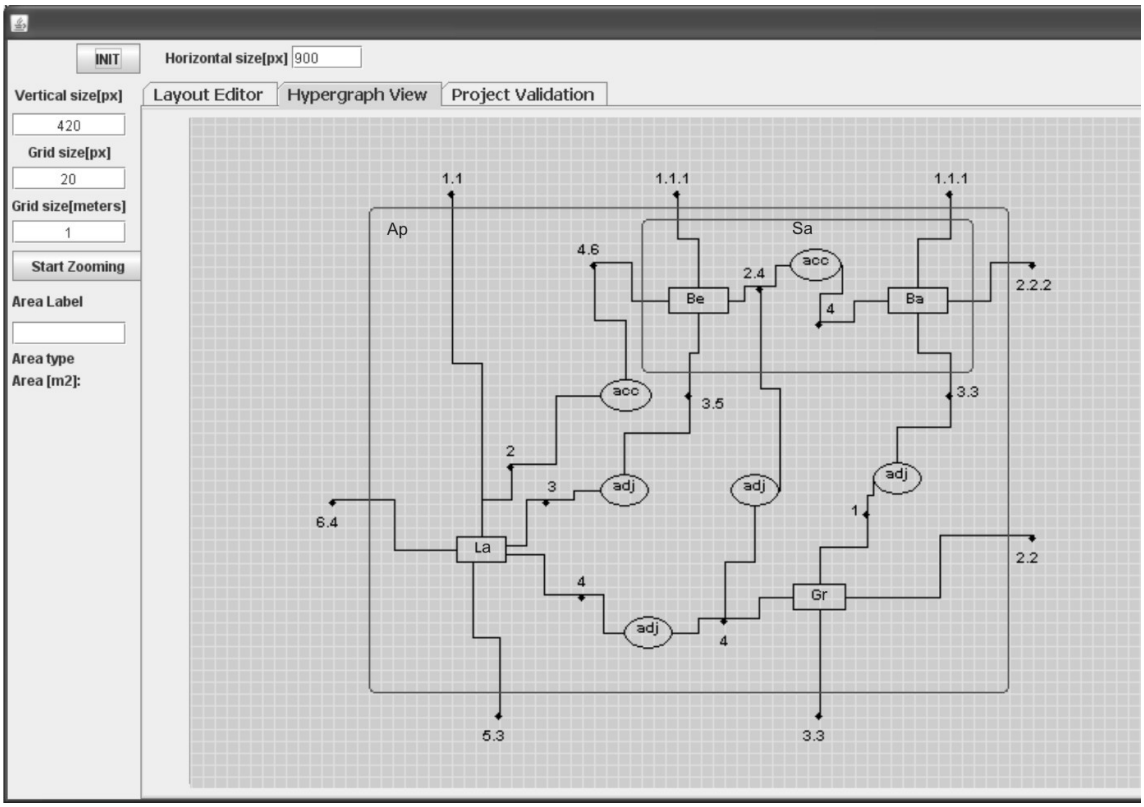
Rys. 4.2. Diagramy projektowe w programie *HSSDR*: a) trzy obszary domu, b) obszar sypialny podzielony na dwa pomieszczenia

Fig. 4.2. Design diagrams in *HSSDR* program: a) three house areas, b) the sleeping area divided into two rooms

Po zastosowaniu opisanej operacji rozwinięcia hiperkrawędzi, hiperkrawędź reprezentująca obszar domu nie posiada już żadnych wierzchołków docelowych. Ich rola została przejęta przez odpowiednie wierzchołki zagnieżdżonego hipergrafu.

W kolejnym kroku projektant dzieli obszar sypialny na dwa pomieszczenia reprezentujące sypialnię (*Be*) i łazienkę (*Ba*). Rysunek 4.2b przedstawia diagram odpowiadający zmody-

fikowanej strukturze projektu. Modyfikacja ta powoduje zagnieżdżenie hipergrafu rozmieszczenia reprezentującego wspomniane dwa pomieszczenia i relację dostępności między nimi w hiperkrawędzi *Sa* reprezentującej obszar sypialny. W wyniku zastosowania tej operacji rozwinięcia hiperkrawędzi powstaje hierarchiczny hipergraf rozmieszczenia pokazany na rys. 4.3.



Rys. 4.3. Hierarchiczny hipergraf rozmieszczenia otrzymany poprzez zagnieżdżenie nowego hipergrafu w hiperkrawędzi reprezentującej obszar sypialny w programie *HSSDR*

Fig. 4.3. A hierarchical layout hypergraph obtained by nesting a new hypergraph in the hyperedge representing the sleeping area in *HSSDR* program

Sześć wierzchołków zewnętrznych hiperkrawędzi etykietowanej *Sa* zostało odpowiednio zastąpionych siedmioma zewnętrznymi wierzchołkami zagnieżdżonego hipergrafu. Ciągi cyfr przypisane tym siedmiu wierzchołkom zostały skonkatenowane z ciągami przypisanymi zastąpionym przez nie wierzchołkom nadrzędnej hiperkrawędzi. Wierzchołek reprezentujący północną ścianę obszaru sypialnego numerowany 1.1 został zastąpiony dwoma wierzchołkami zewnętrznymi odpowiadającymi północnym ścianom sypialni (*Be*) i łazienki (*Ba*) (oba wierzchołki mają numery 1.1.1). Wierzchołki reprezentujące ściany wschodnie obszaru sypialnego (wierzchołki numerowane 2.2 i 4) zostały zastąpione odpowiednio wierzchołkiem o identyfikatorze 2.2.2 reprezentującym wschodnią ścianę łazienki i wierzchołkiem 2.4 reprezentującym wschodnią ścianę sypialni. Wierzchołki reprezentujące południowe ściany (wierzchołki 3 i 5) zastąpiono wierzchołkami o identyfikatorach 3.3 i 3.5 odpowiadającym południowym ścianom łazienki i sypialni, a wierzchołek 6 wierzchołkiem 4.4 reprezentują-

cym zachodnią ścianę sypialni. Hiperkrawędzie relacyjne, które były połączone z wierzchołkami 3, 4, 5 i 6 hiperkrawędzi *Sa*, zostały zastąpione nowymi hiperkrawędziami relacyjnymi połączonymi odpowiednio z wierzchołkami 3.3, 2.4, 3.5 i 4.6 zagnieżdżonego hipergrafu.

Następnie projektant dzieli obszar rekreacyjny (*La*) na pięć pomieszczeń: salon (*Lr*), kuchnię (*Kt*), hol (*Hl*), przedsionek (*En*) i toaletę (*Wc*) (rys. 2.1b). W wyniku kolejnej operacji rozwinięcia hiperkrawędzi, która zagnieżdża hipergraf rozmieszczenia reprezentujący pięć wspomnianych pomieszczeń i relacje pomiędzy nimi w hiperkrawędzi reprezentującej obszar rekreacyjny, powstaje hierarchiczny hipergraf przedstawiony na rys. 3.2. Sześć wierzchołków zewnętrznych hiperkrawędzi etykietowanej *La* zostało odpowiednio zastąpionych zewnętrznymi wierzchołkami zagnieżdżonego hipergrafu.

Wierzchołek obszaru rekreacyjnego (*La*) numerowany 1.1 został zastąpiony wierzchołkiem 1.1.1 reprezentującym północną ścianę salonu. Wierzchołek obszaru rekreacyjnego numerowany 2 został zastąpiony wierzchołkiem 2.2 reprezentującym wschodnią ścianę salonu. Wierzchołek numerowany 3 – wierzchołkiem 1.3 reprezentującym północną ścianę holu, wierzchołek o numerze 4 – dwoma wierzchołkami 2.4 reprezentującymi odpowiednio wschodnie ściany holu i toalety, natomiast wierzchołek 5.3 został zastąpiony trzema wierzchołkami numerowanymi 3.5.3, które odpowiadają południowym ścianom kuchni, przedsionka i toalety. Wierzchołek numerowany 6.4 jest zastąpiony dwoma wierzchołkami 4.6.4 reprezentującymi zachodnie ściany salonu i kuchni.

Trzy hiperkrawędzie relacyjne, które były połączone z wierzchołkami 2, 3 i 4 hiperkrawędzi etykietowanej *La*, są zastąpione czterema nowymi hiperkrawędziami relacyjnymi. Hiperkrawędź, która była połączona z wierzchołkiem 2 obszaru rekreacyjnego, jest zastąpiona hiperkrawędzią połączoną z wierzchołkiem 2.2 zagnieżdżonego hipergrafu, hiperkrawędź, która była połączona z wierzchołkiem 3 obszaru rekreacyjnego, jest zastąpiona hiperkrawędzią połączoną z wierzchołkiem 1.3 holu. Hiperkrawędź, która była połączona z wierzchołkiem 4 obszaru rekreacyjnego, jest zastąpiona przez dwie hiperkrawędzie relacyjne, z których jedna jest połączona z wierzchołkiem 2.4 holu, a druga z wierzchołkiem 2.4 hiperkrawędzi etykietowanej *Wc* (rys. 3.2). ◊

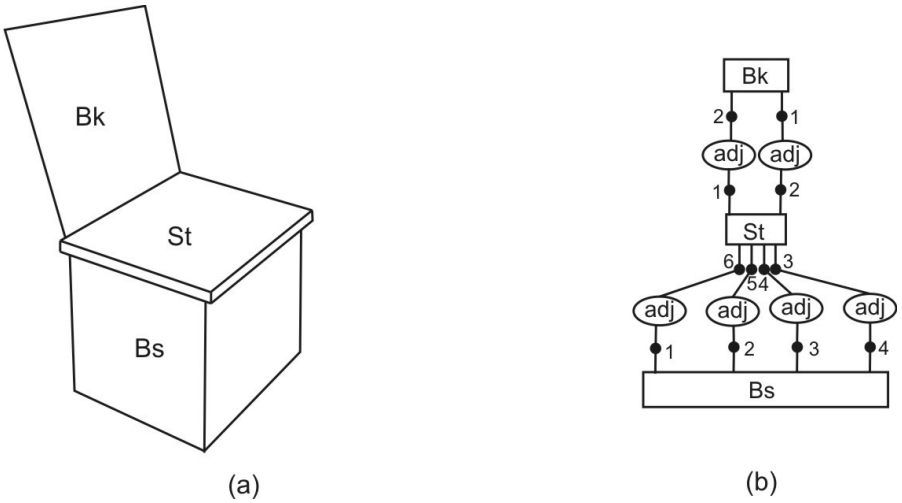
Przykład 4.2: krzesła

Załóżmy, że projektant rozpoczyna tworzenie projektu krzesła od narysowania diagramu reprezentującego jego trzy podstawowe części, a więc siedzisko, oparcie i podstawę (rys. 4.4a). Automatycznie jest generowany początkowy hipergraf reprezentujący strukturę projektu odpowiadającego temu diagramowi (rys. 4.4b). Składa się on z trzech hiperkrawędzi o etykietach *Bk*, *St* i *Bs*, którym są przypisane wierzchołki reprezentujące punkty połączeń pomiędzy poszczególnymi częściami krzesła. Liczba wierzchołków powiązanych z daną hiperkrawędzią reprezentującą obiekt odpowiadający komponentowi diagramu w tym przypadku ustala-

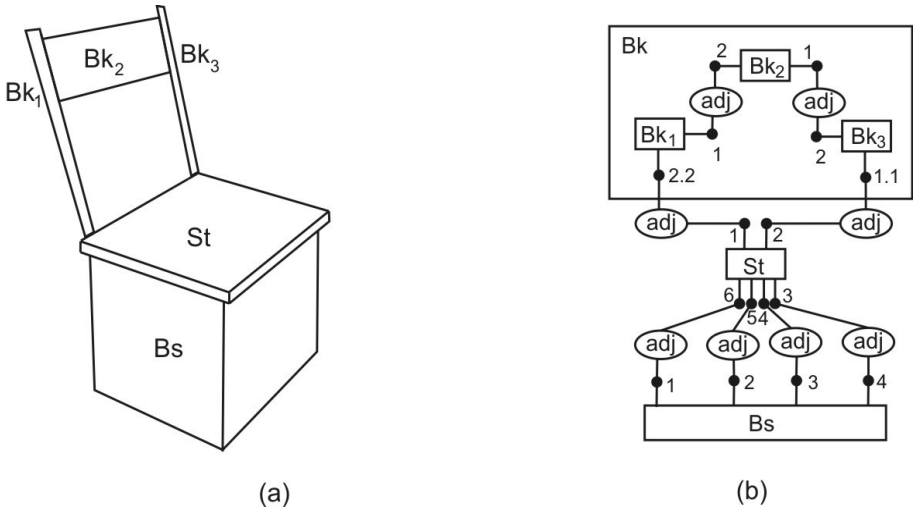
na jest przez projektanta. Wierzchołki te są numerowane zgodnie z ruchem wskazówek zegara, zaczynając od górnego położonego najbardziej na lewo.

W następnym kroku projektant przedstawia oparcie w postaci trzech elementów (rys. 4.5a). Na skutek tej akcji projektowej jest wywoływana operacja rozwinięcia hiperkrawędzi, w wyniku której hipergraf rozmieszczenia reprezentujący wzajemne ułożenie tych trzech elementów zostaje zagnieżdżony w hiperkrawędzi odpowiadającej oparciu. Otrzymany hierarchiczny hipergraf rozmieszczenia jest przedstawiony na rys. 4.5b. Wierzchołki zewnętrzne hiperkrawędzi etykietowanej *Bk* (rys. 4.4b) o numerach 1 i 2 zostają zastąpione dwoma zewnętrznymi wierzchołkami zagnieżdżonego hipergrafu rozmieszczenia, z których pierwszy jest numerowany 1.1 i reprezentuje punkt połączenia trzeciej belki oparcia z siedziskiem, a drugi, numerowany 2.2, reprezentuje punkt połączenia pierwszej belki oparcia z siedziskiem krzesła. Druga część ciągu cyfr będącego wartością atrybutu *order* dla nowych wierzchołków jest dziedziczona po zastąpionych wierzchołkach.

W kolejnym kroku projektant tworzy podstawę krzesła w postaci czterech nóg (rys. 2.2b). Modyfikacja ta powoduje zagnieżdżenie hipergrafu rozmieszczenia reprezentującego te cztery elementy w hiperkrawędzi reprezentującej podstawę. W wyniku zastosowania tej operacji powstaje hierarchiczny hipergraf rozmieszczenia pokazany na rys. 3.3. Cztery wierzchołki zewnętrzne hiperkrawędzi etykietowanej *Bs* zostały odpowiednio zastąpione zewnętrznymi wierzchołkami zagnieżdżonego hipergrafu o numerach 1.1, 1.2, 1.3 i 1.4, z których każdy reprezentuje punkt połączenia jednej nogi krzesła z jego siedziskiem. ◊



Rys. 4.4. Krzesła: a) początkowy diagram projektowy, b) hipergraf rozmieszczenia jego elementów
Fig. 4.4. Chairs: a) an initial design diagram, b) its layout hypergraph



Rys. 4.5. Krzesła: a) drugi krok projektowania, b) hierarchiczny hipergraf rozmieszczenia otrzymany w wyniku zastosowania operacji rozwinięcia hiperkrawędzi reprezentującej oparcie
Fig. 4.5. Chairs: a) a second step in chair designing, b) a hierarchical layout hypergraph obtained as a result of applying the development operation to the hyperedge representing the back

4.2. Operacja usunięcia zawartości hiperkrawędzi

Operacją odwrotną do rozwinięcia hiperkrawędzi jest operacja *usunięcia zawartości hiperkrawędzi*, która została zdefiniowana w [GrŚL08]. Jest ona szczególnie użyteczna w przypadku projektowania złożonych obiektów, ponieważ pozwala rozważać wybrane ich części na wyższym poziomie szczegółowości. Operacja ta odpowiada likwidacji podziału obiektu reprezentowanego przez pewien komponent diagramu i polega na usunięciu hipergrafu zagnieżdżonego wcześniej w jakiejś hiperkrawędzi odpowiadającej składowemu obiektowi projektowanego artefaktu. W wyniku tej operacji wierzchołki zewnętrzne usuwanego hipergrafu są zastępowane nowymi wierzchołkami, które stają się nowymi wierzchołkami docelowymi nadrzędnej hiperkrawędzi. Wierzchołkom tym są przywracane numery jakie miały one przed wykonaniem operacji rozwinięcia hiperkrawędzi.

Definicja 4.2:
Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ będzie hierarchicznym hipergrafem rozmieszczenia nad Σ i A, a $\bar{e} \in E_G^C$ będzie hiperkrawędzią obiektową z G taką, że istnieje hierarchiczny hipergraf rozmieszczenia $H = (E_H, V_H, t_H, s_H, lb_H, att_H, ext_H, ch_H)$ nad Σ i A oraz $ch_G^+(\bar{e}) = E_H \cup V_H$, gdzie ch_G^+ oznacza przechodnie domknięcie ch_G i specyfikuje wszystkie atomy hipergrafu zagnieżdżone w $\bar{e}$. Niech $order \in A$ będzie atrybutem wierzchołkowym ze zbiorem wartości D_{order}.

Niech EXT_H oznacza zbiór wierzchołków zewnętrznych H, V oznacza skończony zbiór wierzchołków taki, że $V \cap V_G = \varnothing$, E oznacza zbiór wszystkich hiperkrawędzi relacyjnych

z $E_G^R - E_H^R$ połączonych z wierzchołkami z EXT_H, a E' oznacza skończony zbiór hiperkrawędzi relacyjnych taki, że $E' \cap E_G^R = \varnothing$.

Niech funkcja *substr*($string_1$, $string_2$) zwraca podciąg ciągu $string_1$ bez prefiksu $string_2$.

Operacja usunięcia zawartości hiperkrawędzi jest zdefiniowana za pomocą trzech funkcji:

1. funkcja usunięcia $sup: EXT_H \to V$ zdefiniowana w taki sposób, że $\forall\, v, w \in EXT_H$ $substr(order_G(v), |order_H(v)|) = substr(order_G(w), |order_H(w)|) \Rightarrow sup(v) = sup(w)$, jest surjekcją wyznaczającą odpowiedniość pomiędzy zbiorem nowych wierzchołków i wierzchołkami zewnętrznymi H, przypisując jeden wierzchołek wszystkim wierzchołkom, których numery będące ciągami cyfr reprezentującymi wartości atrybutu *order* różnią się tylko prefiksem o długości wyznaczonej przez atrybutowanie w H;
2. funkcja $Ev_{order}: V \times order \to D_{order}$ przypisuje wartości atrybutowi *order* wierzchołków z V w taki sposób, że $\forall\, v \in V$, gdzie $v = sup(w)$, $order(v) = substr(order_G(w), |order_H(w)|)$, tzn. nowym wierzchołkom są przypisywane ciągi cyfr otrzymane poprzez usunięcie prefiksów będących numerami odpowiadających im wierzchołków zewnętrznych H;
3. funkcja osadzenia usunięcia $emb_{sup}: E \to E'$ jest surjekcją wyznaczającą odpowiedniość pomiędzy zbiorem nowych hiperkrawędzi relacyjnych i hiperkrawędziami relacyjnymi połączonymi z wierzchołkami zewnętrznymi H.

Wynikiem operacji usunięcia zawartości hiperkrawędzi jest atrybutowany hierarchiczny hipergraf rozmieszczenia $K = (E_K, V_K, t_K, s_K, lb_K, att_K, ext_K, ch_K)$ nad Σ i A, gdzie:

1. $E_K = ((E_G - E_H) - E) \cup E'$,
2. $V_K = (V_G - V_H) \cup V$,
3. $t_K: E_K \to V_K^*$ i $s_K: E_K^R \to V_K^*$ są zdefiniowane w taki sposób, że:
   - $t_K(\bar{e}) \subseteq (T_G(\bar{e}) \cup V)^*$,
   - $\forall\, e \in ((E_G - E_H) - E - \{\bar{e}\})\ t_K(e) = t_G(e)$,
   - $\forall\, e \in ((E_G - E_H) - E) \cap E_G^R\ s_K(e) = s_G(e)$,
   - $\forall\, e \in E'\ s_K(e) = f(e) \wedge t_K(e) = g(e)$, gdzie $f, g: E' \to V_K^*$,
4. lb_K jest zdefiniowane następująco:
   - $\forall\, e \in ((E_G - E_H) - E)\ lb_K(e) = lb_G(e)$,
   - $\forall\, e \in E'$ gdzie $e = emb_{sup}(e_1)$, $lb_K(e) = lb_G(e_1)$,
   - $\forall\, v \in (V_G - V_H)\ lb_K(v) = lb_G(v)$,
   - $\forall\, v \in V$ gdzie $v = sup(w)$, $lb_K(v) = lb_H(w)$,
5. att_K jest zdefiniowane następująco:
   - $\forall\, e \in (E_G^C - E_H^C)\ att_K(e) = att_G(e)$,

- $\forall\ v \in (V_G - V_H)\ att_K(v) = att_G(v)$,
- $\forall\ v \in V\ att_K(v) = h(v)$, gdzie h: $V \rightarrow P(A)$ i $h(v) \subseteq att_H(w)$, dla $v = sup(w)$,

6. ext_K: $[n_K] \rightarrow V_K^*$,
7. ch_K jest zdefiniowane następująco:
   - $ch_K(\bar{e}) = \varnothing$,
   - $\forall\ e \in ((E_G^C - E_H^C) - \{\bar{e}\})\ ch_K(e) = ch_G(e)$. □

W wyniku zastosowania operacji usunięcia zawartości hiperkrawędzi hipergraf H zostaje usunięty z hiperkrawędzi $\bar{e}$ grafu G, która przestaje być hiperkrawędzią hierarchiczną. Wierzchołki zewnętrzne H są zastępowane odpowiadającymi im wierzchołkami ze zbioru V, które stają się nowymi wierzchołkami docelowymi hiperkrawędzi $\bar{e}$. Odpowiedniość pomiędzy tymi wierzchołkami jest wyznaczana przez funkcję usunięcia *sup* na podstawie numerów wierzchołków będących ciągami cyfr reprezentującymi wartości atrybutu *order*. Wierzchołki docelowe hiperkrawędzi $\bar{e}$ otrzymują takie same ciągi cyfr, jakie posiadały przed operacją rozwinięcia $\bar{e}$.

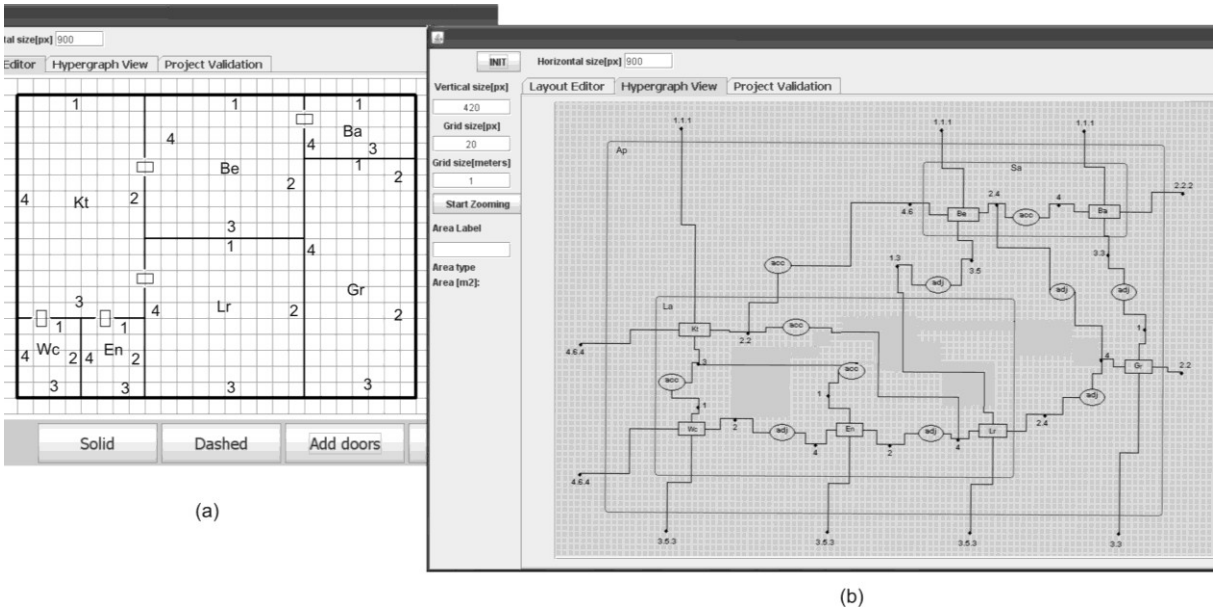
Funkcja osadzenia usunięcia pozwala zastąpić hiperkrawędzie relacyjne połączone z zewnętrznymi wierzchołkami H nowymi hiperkrawędziami relacyjnymi łączącymi wierzchołki, które pozostały w G z wierzchołkami docelowymi $\bar{e}$. Sposób zastąpienia każdej hiperkrawędzi relacyjnej odpowiadającą jej nową hiperkrawędzią jest określany na podstawie wymagań projektowych.

Przykład 4.3: rozkłady pomieszczeń

Załóżmy, że projektant nie jest zadowolony z otrzymanego rozkładu obszaru rekreacyjnego i decyduje się go zmienić. W tym celu usuwa podział tego obszaru i powraca do struktury projektu odpowiadającej diagramowi z rys. 4.2b. Konsekwencją tej akcji projektowej jest automatyczne wykonanie operacji usunięcia zawartości odpowiedniej hiperkrawędzi w aktualnym hierarchicznym hipergrafie rozmieszczenia. W wyniku tej operacji hipergraf zagnieżdżony w hiperkrawędzi *La* reprezentującej obszar rekreacyjny jest usuwany i ponownie powstaje hipergraf z rys. 4.3. Funkcja *sup* operacji usunięcia zawartości hiperkrawędzi określająca sposób zastępowania wierzchołków wyznacza sześć nowych wierzchołków (zbiór V) odpowiadających dziesięciu zewnętrznym wierzchołkom zagnieżdżonego hipergrafu. Numery nowych wierzchołków są otrzymywane przez usunięcie prefiksów pochodzących od odpowiadających im zewnętrznych wierzchołków usuwanego hipergrafu (otrzymują takie same numery, jakie posiadały przed operacją rozwinięcia hiperkrawędzi). Nowe wierzchołki stają się wierzchołkami docelowymi hiperkrawędzi o usuwanej zawartości. Przykładowo, dwa wierzchołki o numerach 2.4 reprezentujące odpowiednio wschodnie ściany holu i toalety są zastępowane jednym wierzchołkiem o numerze 4 reprezentującym wschodnią ścianę obszaru *La*, podczas gdy trzy wierzchołki numerowane 3.5.3 są zastępowane jednym wierzchołkiem

o numerze 5.3 reprezentującym południową ścianę obszaru rekreacyjnego. Cztery hiperkrawędzie relacyjne, które były połączone z wierzchołkami o numerach 2.2, 1.3, 2.4 i 2.4 (E = {2.2, 1.3, 2.4, 2.4}), są zastępowane trzema nowymi hiperkrawędziami relacyjnymi (ze zbioru E') połączonymi z wierzchołkami 2, 3 i 4 hiperkrawędzi etykietowanej *La* zgodnie z funkcją osadzenia usunięcia.

W następnym kroku projektowym obszar rekreacyjny jest dzielony na cztery pomieszczenia odpowiadające salonowi, kuchni, przedsionkowi i toalecie. Diagram projektowy otrzymany w wyniku nowego podziału obszaru *La* i hierarchiczny hipergraf rozmieszczenia reprezentujący strukturę projektu odpowiadającą temu diagramowi są przedstawione na rys. 4.6a i 4.6b. ◊



Rys. 4.6. Rozkład pomieszczeń projektowany w programie *HSSDR*: a) nowy diagram projektowy, b) hierarchiczny hipergraf rozmieszczenia reprezentujący strukturę projektu odpowiadającą temu diagramowi

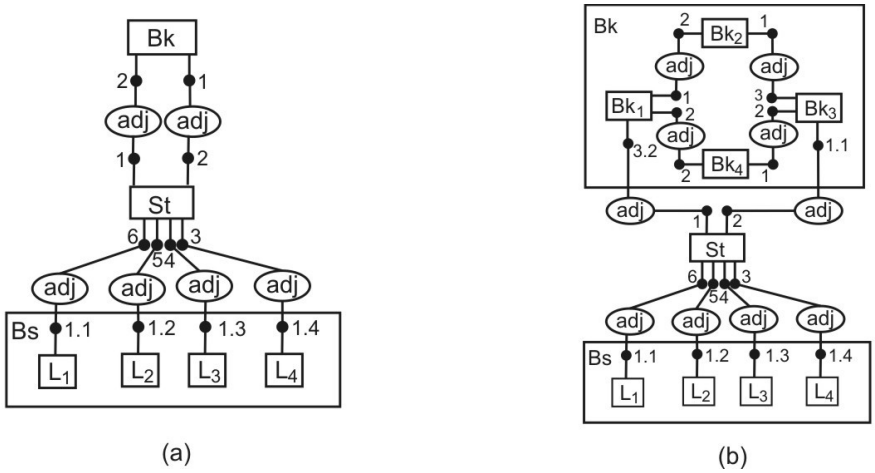
Fig. 4.6. A floor layout designed in *HSSDR* program: a) a new design diagram, b) a hierarchical layout hypergraph representing the project structure corresponding to this diagram

Przykład 4.4: krzesła

Załóżmy, że projektant postanawia zmienić wygląd oparcia krzesła i usuwa z diagramu dotychczasowe jego elementy. Konsekwencją tej akcji projektowej jest wykonanie operacji usunięcia zawartości odpowiedniej hiperkrawędzi aktualnego hierarchicznego hipergrafu rozmieszczenia. W wyniku tej operacji hipergraf zagnieżdżony w hiperkrawędzi *Bk* reprezentującej oparcie jest usuwany i powstaje hipergraf z rys. 4.7a. Dwa zewnętrzne wierzchołki usuwanego hipergrafu zostają zastąpione dwoma nowymi wierzchołkami, które stają się wierzchołkami docelowymi hiperkrawędzi *Bk*. Należy zauważyć, że zmiana wyglądu oparcia nie wymaga cofnięcia procesu projektowego do etapu, na którym poprzednio było ono pro-

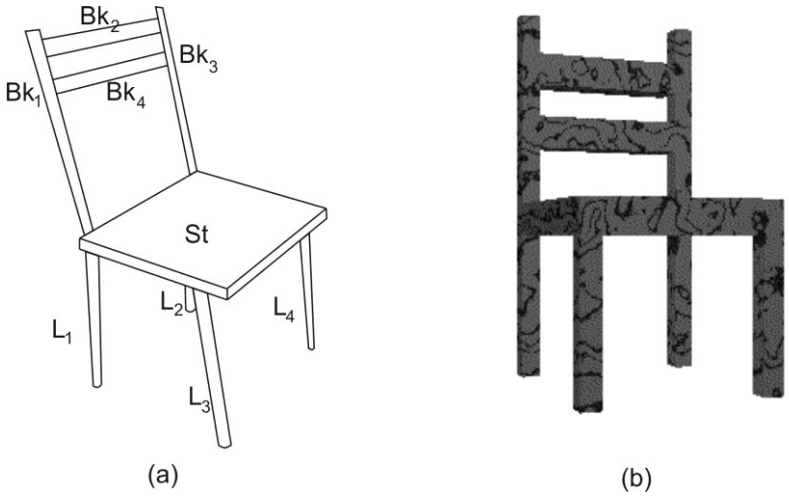
jektowane. Zatem, zarówno wygląd podstawy krzesła, jak i reprezentująca ją cześć hipergrafu pozostają niezmienione.

W następnym kroku oparcie zostaje przedstawione na diagramie w postaci dwóch poziomych i dwóch pionowych belek (rys. 4.8a). Hierarchiczny hipergraf reprezentujący strukturę projektu odpowiadającego nowemu diagramowi jest przedstawiony na rys. 4.7b, a trójwymiarowa wizualizacja tego diagramu na rys. 4.8b. ◊



Rys. 4.7. Hierarchiczne hipergrafy rozmieszczenia: a) po usunięciu zawartości hiperkrawędzi *Bk*, b) odpowiadający nowemu projektowi krzesła

Fig. 4.7. Hierarchical layout hypergraphs: a) after removing the contents of the hyperedge *Bk*, b) corresponding to a new chair design



Rys. 4.8. Krzesła: a) nowy diagram projektowy, b) odpowiadająca mu wizualizacja

Fig. 4.8. Chairs: a) a new design diagram, b) its corresponding visualization

4.3. Operacja konkatenacji hipergrafów

Kolejnymi operacjami na hierarchicznych hipergrafach przydatnymi do odzwierciedlenia w strukturze wewnętrznej zmian wykonywanych przez projektanta na diagramie projekto-

wym są operacja konkatenacji i operacja odłączenia fragmentu hipergrafu. Operacja konkatenacji pozwala dołączyć do istniejącego hierarchicznego hipergrafu nowy hipergraf reprezentujący strukturę nowo dodanego fragmentu projektowanego artefaktu, natomiast operacja odłączenia pozwala usunąć wybrany podgraf hierarchicznego hipergrafu odpowiadający usuwanemu fragmentowi diagramu.

W trakcie projektowania może się okazać, że w strukturze projektu nie została przewidziana odpowiednia liczba obiektów aktywnych umożliwiających wyrażenie relacji pomiędzy nowo dodawanymi obiektami a obiektami już istniejącymi w strukturze. W takim przypadku w hipergrafowej reprezentacji struktury projektu niektórym hiperkrawędziom nie została przypisana wystarczająca liczba wierzchołków, które pozwoliłyby połączyć odpowiednimi hiperkrawędziami relacyjnymi istniejący hipergraf z dołączanym do niego hipergrafem reprezentującym strukturę nowego fragmentu projektu. Dodawaniu do struktury projektu obiektów aktywnych przez projektanta odpowiada operacja zmiany typu hiperkrawędzi, która jest wykonywana jedynie dla hiperkrawędzi obiektowych i polega na zwiększeniu liczby przypisanych im wierzchołków.

Definicja 4.3:
Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ będzie hierarchicznym hipergrafem rozmieszczenia nad Σ i A oraz niech $\bar{e} \in E_G^C$ będzie hiperkrawędzią obiektową z G. Niech V oznacza skończony zbiór wierzchołków taki, że $V \cap V_G = \varnothing$. Niech EXT_G oznacza zbiór wierzchołków zewnętrznych G.

Operacja zmiany typu hiperkrawędzi jest zdefiniowana za pomocą dwóch odwzorowań:

1. odwzorowania add: $\bar{e} \rightarrow (V_G \cup V)^*$ takiego, że $t_G(\bar{e}) \subseteq add(\bar{e})$ i $|add(\bar{e})| > |t_G(\bar{e})|$, przypisującego nowe wierzchołki docelowe hiperkrawędzi $\bar{e}$,
2. funkcji przypisującej wartości atrybutowi $order \in A$ dla wierzchołków z V Ev_{order}: $V \times order \rightarrow [|t_G(\bar{e})|+1, |add(\bar{e})|]$.

Wynikiem operacji zmiany typu hiperkrawędzi jest atrybutowany hierarchiczny hipergraf rozmieszczenia $K = (E_K, V_K, t_K, s_K, lb_K, att_K, ext_K, ch_K)$ nad $\Sigma = O \cup R$ i A, gdzie:

1. $E_K = E_G$,
2. $V_K = V_G \cup V$,
3. t_K: $E_K \rightarrow V_K^*$ i s_K: $E_K^R \rightarrow V_K^*$ i są zdefiniowane w taki sposób, że:
   - $t_K(\bar{e}) = add(\bar{e})$,
   - $\forall\ e \in (E_G - \{\bar{e}\})\ t_K(e) = t_G(e)$,
   - $\forall\ e \in E_G^R\ s_K(e) = s_G(e)$,

4. lb_K jest zdefiniowane następująco:
   - $\forall\ e \in E_G\ lb_K(e) = lb_G(e)$,
   - $\forall\ v \in V_G\ lb_K(v) = lb_G(v)$,
   - $\forall\ v \in V\ lb_K(v) = f(v)$, gdzie f: $V \rightarrow O$,
5. att_K jest zdefiniowane następująco:
   - $\forall\ e \in E_G^C\ att_K(e) = att_G(e)$,
   - $\forall\ v \in V_G\ att_K(v) = att_G(v)$,
   - $\forall\ v \in V\ att_K(v) = g(v)$, gdzie g: $V \rightarrow P(A)$,
6. ext_K: $[n_G + |add(\bar{e})| - |t_G(\bar{e})|] \rightarrow V_K^*$, wyznacza zbiór wierzchołków zewnętrznych K w taki sposób, że $EXT_K = EXT_G \cup V$,
7. ch_K jest zdefiniowane następująco:
   - dla e': $\bar{e} \in ch_G(e')$, $ch_K(e') = ch_G(e') \cup V$,
   - $\forall\ e \in E_G^C$: $\bar{e} \notin ch_G(e)$, $ch_K(e) = ch_G(e)$. □

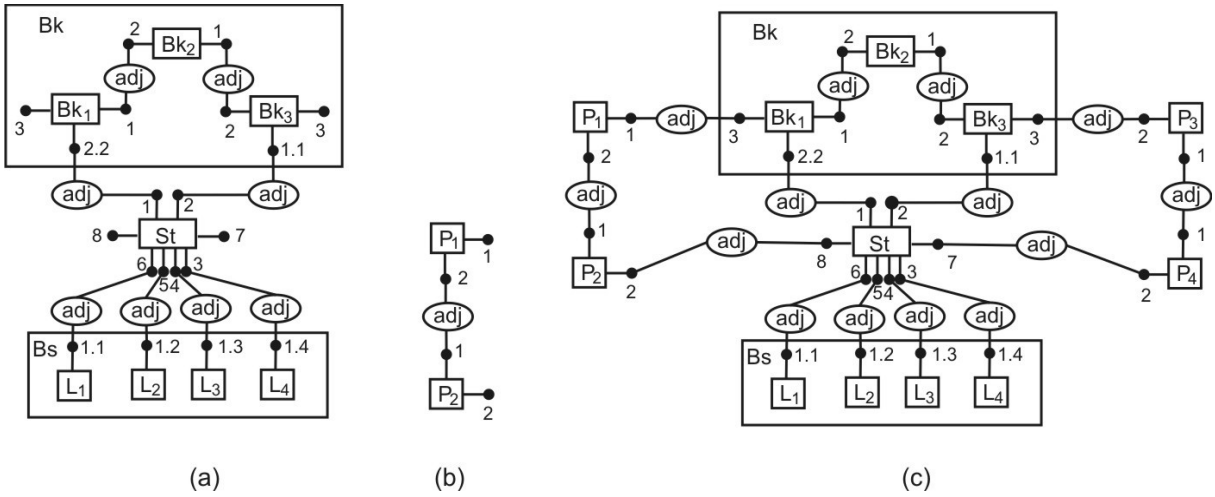
W wyniku zastosowania operacji zmiany typu, hiperkrawędzi $\bar{e}$ hipergrafu G oprócz dotychczasowych wierzchołków docelowych zostają przypisane nowe wierzchołki. Wierzchołki te wydłużają sekwencję wierzchołków docelowych, a więc funkcja przypisująca wartości atrybutowi *order* nadaje im numery z przedziału $[|t_G(\bar{e})|+1, |add(\bar{e})|]$. Nowe wierzchołki stają się także wierzchołkami zewnętrznymi całego hipergrafu, co powoduje zmianę jego typu. Ponadto, wierzchołki te muszą być zagnieżdżone w tej samej hiperkrawędzi co hiperkrawędź zmieniająca swój typ.

Przykład 4.5: krzesła

Załóżmy, że projektant chciałby dodać do projektowanego krzesła poręcze. Najpierw musi on do struktury projektu dodać dwa nowe obiekty odpowiadające punktom połączenia obu belek pionowych oparcia z poręczą oraz dwa obiekty odpowiadające punktom siedziska, w których będą się z nim stykały poręcze krzesła. Konsekwencją takiej zmiany struktury jest automatyczne wykonanie na hipergrafie trzech operacji zmiany typu hiperkrawędzi. Pierwsza z nich dodaje nowy wierzchołek i przypisuje go hiperkrawędzi Bk_1, druga dodaje nowy wierzchołek i przypisuje go hiperkrawędzi Bk_3 (wierzchołki o numerze 3), a trzecia dodaje dwa nowe wierzchołki do hipergrafu i przypisuje je hiperkrawędzi *St* (wierzchołki 7 i 8). W wyniku tych operacji powstaje hipergraf przedstawiony na rys. 4.9a. ◊

Kolejną operacją na hipergrafach jest operacja konkatenacji, która odzwierciedla dodawanie do istniejącej struktury projektu nowych obiektów reprezentujących kolejne fragmenty projektowanego artefaktu. Operacja ta łączy dwa hierarchiczne hipergrafy rozmieszczenia reprezentujące struktury różnych fragmentów artefaktu poprzez dodanie nowych hiperkrawędzi relacyjnych łączących wierzchołki zewnętrzne tych hipergrafów i reprezentujących rela-

cje pomiędzy odpowiednimi fragmentami. Niektóre wierzchołki zewnętrzne łączonych hipergrafów stają się wierzchołkami źródłowymi lub docelowymi nowych hiperkrawędzi relacyjnych, podczas gdy inne z tych wierzchołków mogą stać się dodatkowymi wierzchołkami źródłowymi lub docelowymi hiperkrawędzi relacyjnych istniejących już w drugim z hipergrafów. Ponadto, pewne hiperkrawędzie i wierzchołki jednego z konkatenowanych hipergrafów mogą zostać zagnieżdżone w hiperkrawędziach obiektowych drugiego z hipergrafów.



Rys. 4.9. Hierarchiczne hipergrafy rozmieszczenia: a) hipergraf reprezentujący strukturę krzesła po wykonaniu trzech operacji zmiany typu hiperkrawędzi, b) hipergraf reprezentujący belki poręczy, c) hipergraf z rys 4.9a po dokonkatenowaniu dwóch hipergrafów reprezentujących poręcze krzesła

Fig. 4.9. Hierarchical layout hypergraphs: a) the hypergraph representing the chair structure after the application of three operations changing types of hyperedges, b) a hypergraph representing arm elements, c) the hypergraph from fig. 4.9a after the concatenation of two hypergraphs representing chair arms

Definicja 4.4:

Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ i $H = (E_H, V_H, t_H, s_H, lb_H, att_H, ext_H, ch_H)$ będą dwoma hierarchicznymi hipergrafami rozmieszczenia nad Σ i A oraz niech E będzie skończonym zbiorem hiperkrawędzi relacyjnych, takim że $E \cap E_G = \varnothing$ i $E \cap E_H = \varnothing$. Niech $E_1 \subseteq E_G^R$ i $E_2 \subseteq E_H^R$ będą podzbiorami hiperkrawędzi relacyjnych, odpowiednio hipergrafów G i H, których sekwencje wierzchołków źródłowych lub docelowych zostaną zmienione w wyniku konkatenacji.

Niech EXT_G i EXT_H oznaczają zbiory wierzchołków zewnętrznych hipergrafów G i H.

Operacja konkatenacji dwóch hipergrafów jest zdefiniowana za pomocą następujących odwzorowań:

1. $concs$, $conct$: $E \to (EXT_G \cup EXT_H)^*$, przypisujących nowym hiperkrawędziom relacyjnym z E odpowiednio sekwencje wierzchołków źródłowych i docelowych,
2. $concs_1$, $conct_1$: $E_1 \to EXT_H{}^*$ oraz $concs_2$, $conct_2$: $E_2 \to EXT_G{}^*$ przypisujących istniejącym hiperkrawędziom relacyjnym z E_1 odpowiednio sekwencje dodatkowych wierzchołków źródłowych i docelowych z EXT_H, a hiperkrawędziom relacyjnym z E_2 odpowiednio sekwencje dodatkowych wierzchołków źródłowych i docelowych z EXT_G,
3. par: $E_H \cup V_H \to E_G^C \cup \perp$, przypisującego atomom hipergrafu H rodziców ze zbioru hiperkrawędzi hipergrafu G reprezentujących obiekty, w taki sposób, że $\forall\, e \in E_H^C$ $\forall\, v \in T(e)$ $par(e) = par(v)$, gdzie $T(e)$ oznacza zbiór wszystkich wierzchołków docelowych krawędzi e, czyli wszystkie wierzchołki powiązane z hiperkrawędzią obiektową e z H będą zagnieżdżone w tej samej hiperkrawędzi hipergrafu G co e. Symbol $\perp$ jest stosowany w przypadku braku rodzica dla danego atomu hipergrafu.

Wynikiem operacji konkatenacji hipergrafów G i H jest atrybutowany hierarchiczny hipergraf rozmieszczenia $K = (E_K, V_K, t_K, s_K, lb_K, att_K, ext_K, ch_K)$ nad $\Sigma = O \cup R$ i A, gdzie:

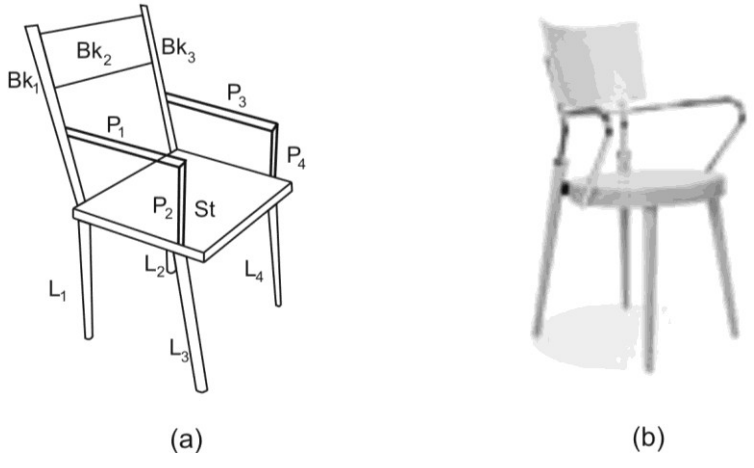
1. $E_K = E_G \cup E_H \cup E$,
2. $V_K = V_G \cup V_H$,
3. t_K: $E_K \to V_K{}^*$ i s_K: $E_K^R \to V_K{}^*$ i są zdefiniowane w taki sposób, że:
   - $\forall\, e \in E\ s_K(e) = concs(e) \wedge t_K(e) = conct(e)$,
   - $\forall\, e \in (E_G - E_1)\ t_K(e) = t_G(e)$,
   - $\forall\, e \in (E_G^R - E_1)\ s_K(e) = s_G(e)$,
   - $\forall\, e \in E_1\ s_K(e) = s_G(e) + concs_1(e) \wedge t_K(e) = t_G(e) + conct_1(e)$,
   - $\forall\, e \in (E_H - E_2)\ t_K(e) = t_H(e)$,
   - $\forall\, e \in (E_H^R - E_2)\ s_K(e) = s_H(e)$,
   - $\forall\, e \in E_2\ s_K(e) = s_H(e) + concs_2(e) \wedge t_K(e) = t_H(e) + conct_2(e)$,
4. lb_K jest zdefiniowane następująco:
   - $\forall\, e \in E_G\ lb_K(e) = lb_G(e)$,
   - $\forall\, e \in E_H\ lb_K(e) = lb_H(e)$,
   - $\forall\, e \in E\ lb_K(e) = f(e)$, gdzie f: $E \to R$,
   - $\forall\, v \in V_G\ lb_K(v) = lb_G(v)$,
   - $\forall\, v \in V_H\ lb_K(v) = lb_H(v)$,
5. att_K jest zdefiniowane następująco:
   - $\forall\, e \in E_G^C\ att_K(e) = att_G(e)$,
   - $\forall\, e \in E_H^C\ att_K(e) = att_H(e)$,

- $\forall\ v \in V_G\ att_K(v) = att_G(v)$,
- $\forall\ v \in V_H\ att_K(v) = att_H(v)$,

6. ext_K: $[n_K] \rightarrow V_K^*$,
7. ch_K jest zdefiniowane następująco:
   - $\forall\ e \in E_G^C\ ch_K(e) = ch_G(e) \cup \{e' \in E_H \mid par(e') = e\} \cup \{v' \in V_H \mid par(v') = e\}$,
   - $\forall\ e \in E_H^C\ ch_K(e) = ch_H(e)$. □

Przykład 4.6: krzesła

Wróćmy do poprzednio rozważanego przykładu dodawania poręczy do krzesła. Po dorysowaniu do diagramu pierwszej poręczy składającej się z dwóch belek automatycznie jest wykonywana operacja konkatenacji hipergrafu z rys. 4.9a z hipergrafem reprezentującym połączone ze sobą belki poręczy (rys. 4.9b). W wyniku tej operacji zostają dodane dwie nowe hiperkrawędzie relacyjne, z których jedna łączy wierzchołek 3 przypisany hiperkrawędzi Bk_1 reprezentujący punkt połączenia belki oparcia z wierzchołkiem 1 hiperkrawędzi P_1 reprezentującym punkt połączenia poziomej belki poręczy, natomiast druga łączy wierzchołek 8 hiperkrawędzi St reprezentujący punkt połączenia siedziska z wierzchołkiem 2 hiperkrawędzi P_2 reprezentującym punkt połączenia pionowej belki poręczy (rys. 4.9c). Po dorysowaniu drugiej poręczy otrzymany hipergraf jest konkatenowany z hipergrafem reprezentującym drugą poręcz w analogiczny sposób. Hierarchiczny hipergraf otrzymany po wykonaniu obu operacji konkatenacji jest pokazany na rys. 4.9c. Diagram projektowy krzesła z dwoma poręczami widać na rys. 4.10a, a jego wizualizację na rys. 4.10b. ◊

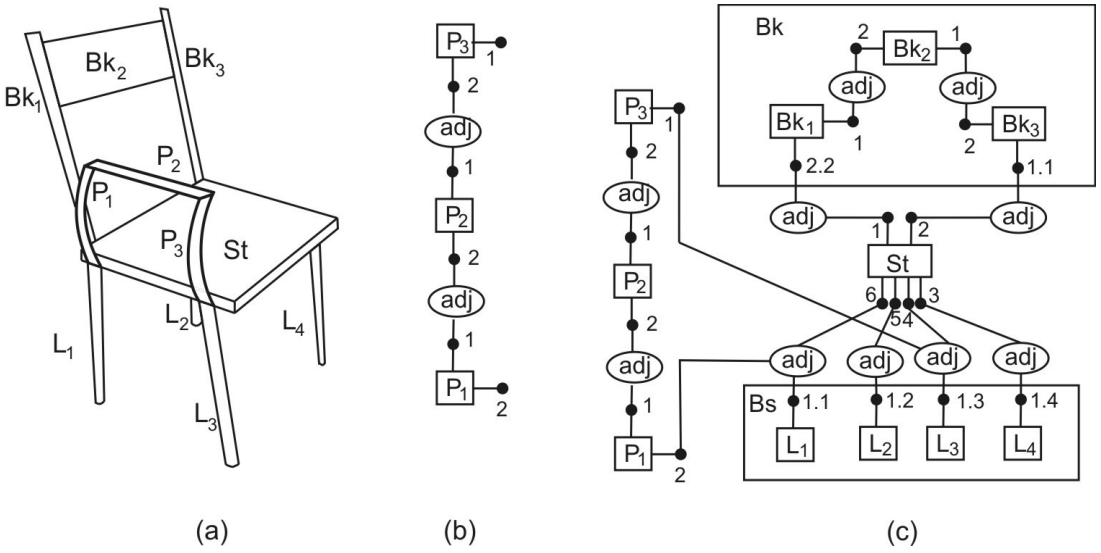


Rys. 4.10. Krzesła: a) kolejny diagram projektowy, b) odpowiadająca mu wizualizacja
Fig. 4.10. Chairs: a) the next design diagram, b) its visualization

Przykład 4.7: krzesła

Załóżmy, że projektant rozważa inną konstrukcję krzesła z dwoma poręczami. Poręcze zostaną przymocowane od spodniej strony siedziska w tych samych miejscach co nogi krzesła.

Nowy diagram projektowy takiego krzesła jest pokazany na rys. 4.11a. W tym przypadku punkty reprezentujące nowe połączenia siedziska nie są potrzebne. Nie będzie więc wywoływana operacja zmiany typu hiperkrawędzi. Po zmianie diagramu od razu zostanie wykonywana operacja konkatenacji hipergrafu z rys. 3.3 z hipergrafem reprezentującym trzy połączone ze sobą części poręczy (rys. 4.11b). W celu konkatenacji tych hipergrafów nie są dodawane żadne nowe hiperkrawędzie relacyjne, a jedynie istniejącym już dwóm hiperkrawędziom reprezentującym przyleganie nóg krzesła (L_1, L_3) do siedziska zostają przypisane nowe wierzchołki reprezentujące punkty połączenia odpowiednich fragmentów poręczy (wierzchołki 2 i 1 hiperkrawędzi P_1 i P_3). Otrzymany hipergraf jest pokazany na rys. 4.11c. ◊



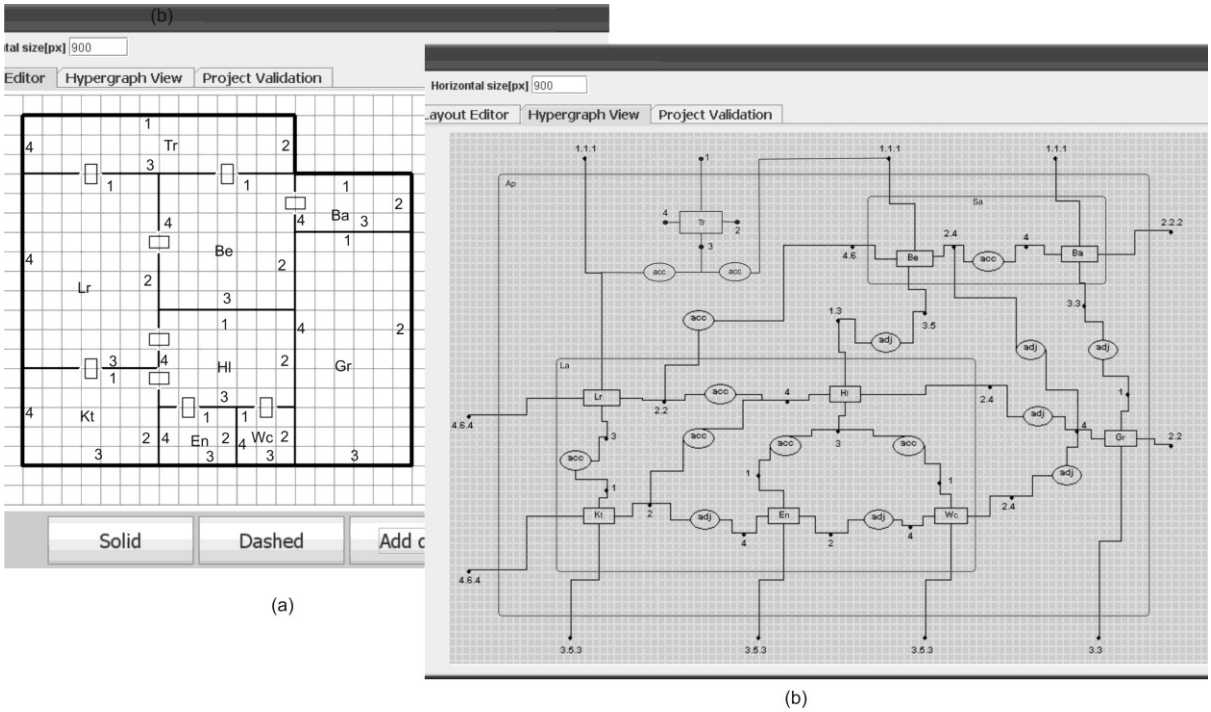
Rys. 4.11. Krzesła: a) diagram krzesła z trzyczęściową poręczą, b) hipergraf reprezentujący trzy części poręczy, c) hipergraf reprezentujący strukturę odpowiadającą diagramowi z rys. 4.11a

Fig. 4.11. Chairs: a) a diagram of a chair with a three-element arm, b) a hypergraph representing three parts of the arm, c) a hypergraph representing the structure corresponding to the diagram from fig. 4.11a

Przykład 4.8: rozkłady pomieszczeń

Wróćmy do przykładu projektowania rozkładu pomieszczeń. Załóżmy, że do zaprojektowanego już rozkładu przedstawionego na diagramie z rys. 2.1b projektant chce dodać taras od strony północnej, który byłby dostępny zarówno z salonu, jak i z sypialni. Po dorysowaniu do diagramu tarasu (rys. 4.12a) zostanie wykonana operacja konkatenacji hipergrafu z rys. 3.2 z hipergrafem złożonym z jednej hiperkrawędzi etykietowanej *Tr* i reprezentującej taras oraz czterech wierzchołków przypisanych tej hiperkrawędzi reprezentujących cztery boki tarasu. W wyniku konkatenacji zostają dodane dwie nowe hiperkrawędzie relacyjne reprezentujące dostępność, które łączą wierzchołek 3 tarasu odpowiadający południowemu bokowi odpowiednio z wierzchołkami numerowanymi 1.1.1 reprezentującymi północne ściany salonu

i sypialni. Dodatkowo hiperkrawędź etykietowana *Tr* wraz z przypisanymi jej wierzchołkami zostaje zagnieżdżona w hiperkrawędzi etykietowanej *Ap* reprezentującej cały obszar mieszkalny. Wynikowy hierarchiczny hipergraf jest przedstawiony na rys. 4.12b. ◊



Rys. 4.12. Rozkład pomieszczeń w programie *HSSDR*: a) diagram rozkładu pomieszczeń z tarasem, b) hipergraf otrzymany w wyniku konkatenacji i reprezentujący strukturę odpowiadającą temu diagramowi

Fig. 4.12. A floor layout in *HSSDR* program: a) a diagram of the floor layout with a terrace obtained, b) a hypergraph obtained as a result of concatenation and representing the structure corresponding to the diagram

4.4. Operacja usunięcia podgrafu hipergrafu

Operacją odwrotną do konkatenacji jest operacja odłączenia odpowiadająca usunięciu pewnego fragmentu diagramu. Pozwala ona usunąć pewien podgraf hierarchicznego hipergrafu reprezentujący wybrany fragment projektowanego artefaktu. Przed zdefiniowaniem operacji odłączenia zostanie zdefiniowany podgraf danego hipergrafu. Podgraf taki składa się z wybranych uchwytów (def. 3.2) oraz hiperkrawędzi relacyjnych łączących wierzchołki zewnętrzne tych uchwytów.

Niech $S_G(e)$ i $T_G(e)$ oznaczają odpowiednio zbiory wierzchołków źródłowych i docelowych wyznaczonych przez odwzorowania s_G i t_G w hipergrafie G.

Definicja 4.5:

Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ i $H = (E_H, V_H, t_H, s_H, lb_H, att_H, ext_H, ch_H)$ będą dwoma hierarchicznymi hipergrafami rozmieszczenia nad Σ i A.

H* jest podgrafem *G ($H \subseteq G$), jeśli spełnione są następujące warunki:

- $E_H \subseteq E_G$, $V_H \subseteq V_G$ i $V_H = \sum_{e \in E_H^C} T_G(e)$, czyli zbiór wierzchołków podgrafu H składa się tylko z wierzchołków przypisanych hiperkrawędziom obiektowym H,
- $\forall\, e \in E_H^R \;\; \exists\, v_1, v_2 \in S_G(e) \cup T_G(e) \mid v_1, v_2 \in V_H,\; v_1 \neq v_2$, czyli każda hiperkrawędź relacyjna podgrafu H łączy co najmniej dwa różne wierzchołki należące do podgrafu,
- $\forall\, e \in E_H^C \;\; ch^+(e) \subset P(E_H \cup V_H)$, czyli wszystkie atomy zagnieżdżone w hierarchicznej hiperkrawędzi podgrafu H też należą do tego podgrafu,
- $t_H = t_G|_{E_H}$ i $s_H = s_G|_{E_H^R}$,
- $lb_H = lb_G|_{E_H \cup V_H}$, $att_H = att_G|_{E_H^C \cup V_H}$, $ch_H = ch_G|_{E_H^C}$,
- ext_H: $[n_H] \rightarrow V_H^*$. □

Operacja usuwająca wybrany podgraf danego hipergrafu powoduje zmianę sekwencji wierzchołków źródłowych i docelowych pewnych relacyjnych hiperkrawędzi reszty hipergrafu, gdyż niektóre z tych wierzchołków mogą zostać usunięte jako należące do podgrafu. Dlatego oprócz wybranego podgrafu zostaną także usunięte skierowane hiperkrawędzie relacyjne, które w sekwencjach przypisanych im wierzchołków źródłowych i docelowych nie zawierają co najmniej po jednym wierzchołku niepołączonym z hiperkrawędziami obiektowymi usuwanego hipergrafu, oraz nieskierowane hiperkrawędzie relacyjne, które w sekwencjach przypisanych im wierzchołków docelowych nie posiadają przynajmniej dwóch wierzchołków nienależących do usuwanego podgrafu.

Definicja 4.6:
Niech $G = (E_G, V_G, t_G, s_G, lb_G, att_G, ext_G, ch_G)$ i $H = (E_H, V_H, t_H, s_H, lb_H, att_H, ext_H, ch_H)$ będą dwoma hierarchicznymi hipergrafami rozmieszczenia nad Σ i A oraz niech H będzie podgrafem G. Niech $E_2 \subseteq E_G^R - E_H^R$ będzie podzbiorem hiperkrawędzi relacyjnych, których sekwencje wierzchołków źródłowych lub docelowych zostaną zmienione w wyniku usunięcia H, czyli takich, że $\forall\, e \in E_2 \;\; \exists\, v_1 \in S_G(e) \cap V_H \vee \exists\, v_2 \in T_G(e) \cap V_H$. Niech $E_1 \subseteq E_2$ będzie podzbiorem hiperkrawędzi relacyjnych, które zostaną usunięte z G w wyniku usunięcia podgrafu H, czyli takich, że $\forall\, e \in E_1$: $s(e) \neq \varnothing$, $\neg\exists\, (v_1 \in S_G(e) - V_H \wedge v_2 \in T_G(e) - V_H,\; v_1 \neq v_2)$ oraz $\forall\, e \in E_1$: $s(e) = \varnothing$, $\neg\exists\, (v_1, v_2 \in T_G(e) - V_H, v_1 \neq v_2)$.

Operacja usunięcia podgrafu H hipergrafu G jest zdefiniowana za pomocą odwzorowań *rems*, *remt*: $E_2 - E_1 \rightarrow (V_G - V_H)^*$ przypisujących hiperkrawędziom relacyjnym z $E_2 - E_1$ odpowiednio sekwencje wierzchołków źródłowych i docelowych w taki sam sposób jak odwzorowania s_G i t_G tylko z pominięciem wierzchołków usuwanego podgrafu, tzn. $\forall\, e \in (E_2 - E_1)$ *rems*(e) wyznacza sekwencję wierzchołków źródłowych dla e złożoną z wierzchołków ze

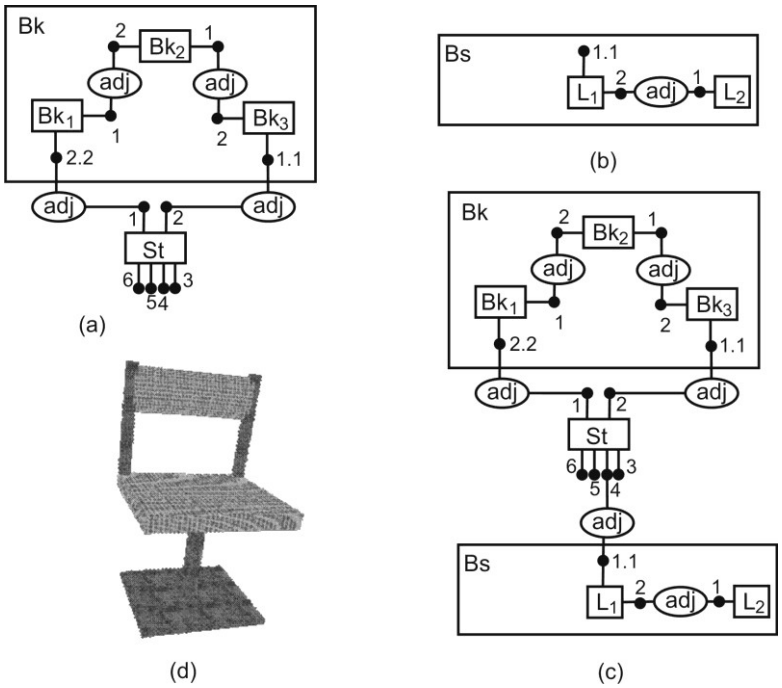
zbioru $S_G(e) - V_H$, a $remt(e)$ wyznacza sekwencję wierzchołków docelowych złożoną z wierzchołków ze zbioru $T_G(e) - V_H$.

Wynikiem operacji usunięcia podgrafu H hipergrafu G jest atrybutowany hierarchiczny hipergraf rozmieszczenia $K = (E_K, V_K, t_K, s_K, lb_K, att_K, ext_K, ch_K)$ nad Σ i A, gdzie:

1. $E_K = E_G - (E_H \cup E_1)$,
2. $V_K = V_G - V_H$,
3. $t_K: E_K \to V_K^*$ i $s_K: E_K^R \to V_K^*$ i są zdefiniowane w taki sposób, że:
   - $\forall\, e \in (E_G - (E_H \cup E_1)) - E_2\ t_K(e) = t_G(e)$,
   - $\forall\, e \in ((E_G - (E_H \cup E_1)) - E_2) \cap E_G^R\ s_K(e) = s_G(e)$,
   - $\forall\, e \in E_2\ s_K(e) = rems(e) \wedge t_K(e) = remt(e)$,
4. $lb_K = lb_G|_{E_K \cup V_K}$,
5. $att_K = att_G|_{E_K^C \cup V_K}$,
6. $ext_K: [n_K] \to V_K^*$,
7. $\forall\, e \in (E_G^C - E_H^C)\ ch_K(e) = ch_G(e) - \{e' \in E_H \cup E_1\} - \{v' \in V_H\}$. □

Przykład 4.9: krzesła

Rozważmy zmianę projektu krzesła z rys. 2.2b polegającą na zastąpieniu czterech nóg jedną nogą składającą się z części pionowej i poziomej. Po usunięciu nóg krzesła przez projektanta zostaje wywołana operacja usunięcia podgrafu z hierarchicznego hipergrafu przedstawionego na rys. 3.3. Podgraf ten składa się z hierarchicznej hiperkrawędzi etykietowanej *Bs* oraz czterech zagnieżdżonych w niej hiperkrawędzi etykietowanych $L_1,\ldots, L_4$ wraz z przypisanymi im wierzchołkami. Wraz z podgrafem zostaną także usunięte cztery hiperkrawędzie relacyjne reprezentujące połączenie nóg z fragmentami siedziska. Otrzymany hipergraf jest pokazany na rys. 4.13a. Po dodaniu do krzesła nowej podstawy zostanie wykonana operacja konkatenacji hipergrafu z rys. 4.13a z hipergrafem złożonym z hierarchicznej hiperkrawędzi etykietowanej *Bs* oraz zagnieżdżonego w niej hipergrafu, którego hiperkrawędzie L_1 i L_2 reprezentują dwie części nowej podstawy, a hiperkrawędź relacyjna reprezentuje połączenie pomiędzy tymi częściami (rys. 4.13b). W wyniku konkatenacji zostanie utworzona nowa hiperkrawędź relacyjna reprezentująca połączenie fragmentu górnej części podstawy z fragmentem siedziska, któremu odpowiada wierzchołek 4. Wartość atrybutu *location* tego wierzchołka musi zostać tak zmodyfikowana, aby określała żądaną lokalizację odpowiadającego mu fragmentu. Hierarchiczny hipergraf otrzymany po opisanej operacji konkatenacji jest pokazany na rys. 4.13c. Model nowego krzesła jest przedstawiony na rys. 4.13d. ◊



Rys. 4.13. Projektowanie krzeseł: a) hierarchiczny hipergraf z rys. 3.3 po operacji usunięcia podgrafu, b) hipergraf reprezentujący nową podstawę krzesła, c) hipergraf otrzymany w wyniku konkatenacji hipergrafów z rys 4.13a i 4.13b, d) model krzesła reprezentowanego przez ten hipergraf

Fig. 4.13. Designing chairs: a) the hierarchical hypergraph from fig. 3.3 after the operation of removing the subgraph, b) a hypergraph representing a new chair base, c) the hypergraph obtained as a result of concatenation of hypegraphs from figs. 4.13a and 4.13b, d) a chair model represented by this hypergraph

4.5. Podsumowanie

W rozdziale tym zostały omówione operacje pozwalające modyfikować hierarchiczne hipergrafy rozmieszczenia reprezentujące struktury projektowanych obiektów. Operacje te odpowiadają poszczególnym akcjom projektowym wykonywanym przez użytkownika na diagramie i zmieniającym jego strukturę. Została zdefiniowana operacja rozwinięcia hiperkrawędzi, która odzwierciedla podział wybranego obiektu na części składowe, i operacja usunięcia zawartości hiperkrawędzi odzwierciedlająca usunięcie podziału wybranego obiektu. Kolejne wyspecyfikowane operacje to zmiana typu hiperkrawędzi odpowiadająca dodawaniu przez projektanta do struktury projektu nowych obiektów aktywnych, operacja konkatenacji hipergrafów wykonywana na skutek dodania nowego fragmentu do projektowanego artefaktu oraz operacja usunięcia podgrafu hierarchicznego hipergrafu odpowiadająca likwidacji wybranej części tego artefaktu. Zostały podane przykłady zastosowań zdefiniowanych operacji.

5. HIERARCHICZNE GRAMATYKI HIPERGRAFOWE

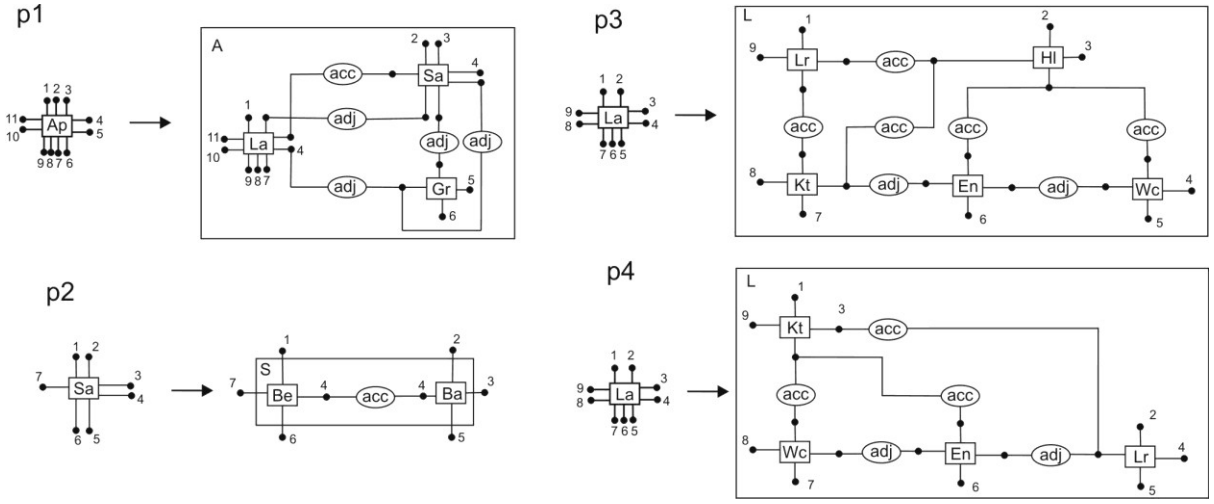
W przypadku gdy struktury projektowanych obiektów są opisywane za pomocą hierarchicznych hipergrafów, efektywnym narzędziem służącym do generacji takich struktur są, oprócz operacji na hipergrafach, hierarchiczne gramatyki hipergrafowe. Zdefiniowana w tym rozdziale hierarchiczna hipergrafowa gramatyka rozmieszczenia jest modyfikacją hierarchicznej gramatyki grafowej opisanej w [Ślus08]. Produkcje tej gramatyki stosowane w kolejnych krokach wywodu pozwalają otrzymać hipergraf rozmieszczenia reprezentujący strukturę rozwiązania danego problemu projektowego. Inteligentni asystenci projektowi, którzy zostaną omówieni w rozdziale 7, wykorzystywani w systemach wspomagania projektowania modelowanych z wykorzystaniem przedstawionej konceptualizacji projektowej i języka wizualnego, generują wewnętrzne reprezentacje projektów z użyciem hierarchicznych gramatyk hipergrafowych. Otrzymane hipergrafy rozmieszczenia są następnie wizualizowane w postaci odpowiednich diagramów z wykorzystaniem odwzorowania zwanego realizacją hipergrafu (def. 3.4).

Konsekwencją stosowania produkcji gramatyki w trakcie wywodu hipergrafu jest wstawianie nowego podgrafu do modyfikowanego hipergrafu, co wymaga określenia transformacji osadzenia, czyli ustalenia połączeń pomiędzy wstawianym podgrafem a resztą modyfikowanego hipergrafu. Ze względu na trudność definiowania transformacji osadzenia najczęściej wprowadza się pewne ograniczenia dotyczące postaci produkcji. Gramatyka hipergrafowa nie jest więc tak elastycznym narzędziem jak operacje na hipergrafach, ale jest szczególnie użyteczna przy projektowaniu obiektów o złożonych strukturach z wielokrotnie powtarzającymi się podstrukturami.

Hierarchiczna hipergrafowa gramatyka rozmieszczenia składa się ze zbioru hiperkrawędzi, zbioru wierzchołków, zbioru produkcji oraz aksjomatu będącego początkowym hipergrafem. Produkcje gramatyki mają postać $p = (l, r, \xi, sr)$, gdzie l i r są atrybutowanymi hierarchicznymi hipergrafami rozmieszczenia posiadającymi tę samą liczbę uporządkowanych wierzchołków zewnętrznych, ξ jest predykatem stosowalności produkcji, a sr jest zbiorem reguł semantycznych określających sposób przypisywania atrybutów hiperkrawędziom i wierzchołkom z r względem atrybutów przypisanych hiperkrawędziom i wierzchołkom z l.

Przykład 5.1: rozkłady pomieszczeń

Wybrane produkcje hierarchicznej hipergrafowej gramatyki rozmieszczenia generującej hipergrafy reprezentujące rozkłady pomieszczeń są przedstawione na rys. 5.1. Dla uproszczenia zostały pominięte predykaty stosowalności produkcji i atrybutowanie hipergrafów. Numery przypisane wierzchołkom hipergrafów lewych i prawych stron produkcji oznaczają kolejne wierzchołki zewnętrzne tych hipergrafów. Po zastosowaniu produkcji $p1$, $p2$ i $p3$ otrzymujemy hipergraf z rys. 3.2, a po zastosowaniu produkcji $p1$, $p2$ i $p4$ – hipergraf z rys. 4.6b. ◊



Rys. 5.1. Produkcje hierarchicznej gramatyki hipergrafowej generującej struktury rozkładów pomieszczeń

Fig. 5.1. Productions of a hierarchical hypergraph grammar generating structures of floor layouts

Niech $\mathcal{AT} = V \cup E$ będzie zbiorem atomów $\mathcal{H}$, gdzie $\mathcal{H}$ oznacza rodzinę atrybutowanych hierarchicznych hipergrafów rozmieszczenia nad Σ i A.

Definicja 5.1:

Hierarchiczna hipergrafowa gramatyka rozmieszczenia nad Σ jest systemem

$$\mathcal{G} = (V, E, P, X),$$

gdzie:

1. V jest skończonym zbiorem wierzchołków,
2. $E = E_C \cup E_R$ jest skończonym zbiorem etykietowanych i atrybutowanych hiperkrawędzi obiektowych i relacyjnych z funkcją zagnieżdżenia ch: $E_C \to P(\mathcal{AT})$,
3. P jest skończonym zbiorem produkcji postaci $p = (l, r, \xi, sr)$ spełniających następujące warunki:
   - l i r są atrybutowanymi hierarchicznymi hipergrafami rozmieszczenia tego samego typu (tzn. $|ext_l| = |ext_r|$) złożonymi z wierzchołków z V i hiperkrawędzi z E,
   - ξ: $\mathcal{H} \to \{TRUE, FALSE\}$ jest predykatem stosowalności produkcji,

- *sr* jest zbiorem reguł semantycznych definiujących wartości atrybutów przypisanych hiperkrawędziom i wierzchołkom z *r*, $sr = \{sr_a \mid sr_a: (E_r \cup V_r) \times A \rightarrow D_a\}$, gdzie $a \in A$ i D_a jest zbiorem wartości *a*,

4. $\mathcal{X}$ jest hipergrafem początkowym nazywanym *aksjomatem* $\mathcal{G}$. □

5.1. Bezkontekstowe hierarchiczne gramatyki hipergrafowe

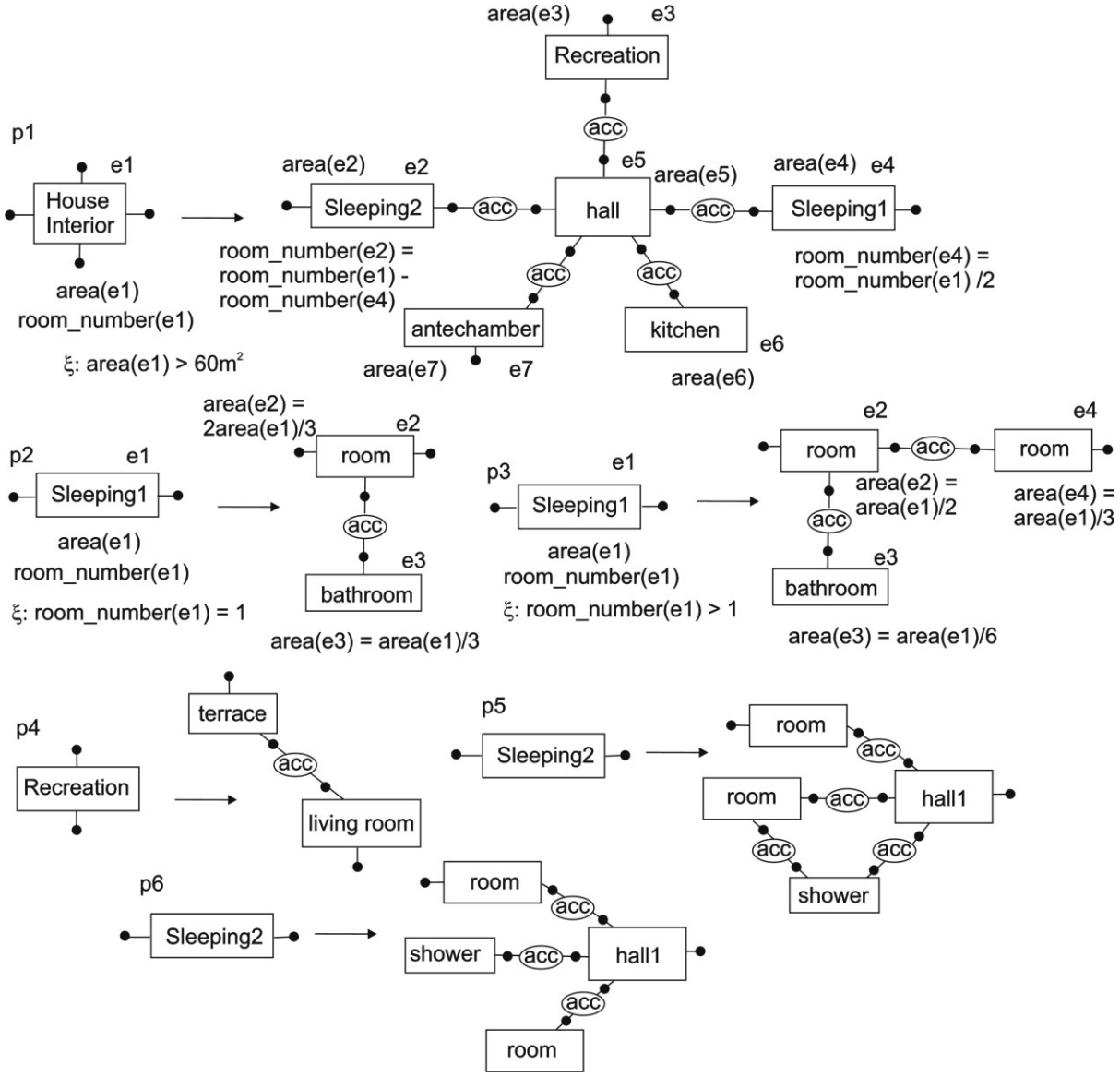
W dalszej części pracy będziemy rozważać jedynie bezkontekstowe hierarchiczne hipergrafowe gramatyki rozmieszczenia, w których lewa strona każdej produkcji jest hipergrafem złożonym z jednej hiperkrawędzi obiektowej o pustej zawartości i zbioru przypisanych jej wierzchołków docelowych będących jednocześnie wierzchołkami zewnętrznymi tego hipergrafu.

Definicja 5.2:
Bezkontekstowa hierarchiczna hipergrafowa gramatyka rozmieszczenia jest hierarchiczną hipergrafową gramatyką rozmieszczenia $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P}, \mathcal{X})$, gdzie:

- lewa strona każdej produkcji $p = (l, r, \xi, sr) \in \mathcal{P}$, jest hipergrafem $l = (E_l, V_l, t_l, s_l, lb_l, att_l, ext_l, ch_l)$ takim, że:
  - $E_l = E_l^C = \{e\}$,
  - $V_l = T(e) = EXT_l$, gdzie $T(e)$ jest zbiorem wierzchołków występujących w sekwencji $t_l(e)$, a EXT_l jest zbiorem wierzchołków zewnętrznych występujących w sekwencji wyznaczonej przez ext_l,
  - $ch_l(e) = \varnothing$,
- $\mathcal{X}$ jest hipergrafem takiej samej postaci jak hipergrafy *l*. □

Przykład 5.2: rozkłady pomieszczeń
Wybrane reguły hierarchicznej hipergrafowej gramatyki rozmieszczenia generującej hipergrafy reprezentujące inne rozkłady pomieszczeń niż gramatyka z przykładu 5.1 są przedstawione na rys. 5.2. Dla pierwszych trzech produkcji przedstawione zostały niektóre atrybuty przypisane hiperkrawędziom reprezentującym obszary i pomieszczenia (atrybuty *area* i *room_number* określające odpowiednio rozmiar obszaru lub pomieszczenia i liczbę pomieszczeń w danym obszarze), niektóre reguły semantyczne i predykaty stosowalności. Wszystkie hiperkrawędzie relacyjne reprezentują dostępność pomiędzy pomieszczeniami. Pierwsza produkcja dzieli obszar mieszkania na trzy obszary funkcjonalne oraz hol, kuchnię i przedpokój, pod warunkiem że dostępny obszar wynosi więcej niż 60 m^2. Druga i trzecia produkcja pozwalają wybrać rozkład pomieszczeń z jednym lub dwoma pokojami w pierwszej części sy-

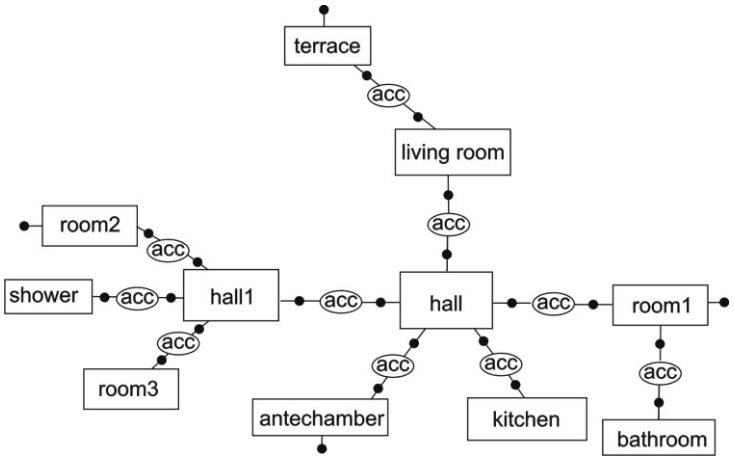
pialnej. Produkcję trzecią można zastosować, jeśli wartość atrybutu *room_number* jest większa niż 1.



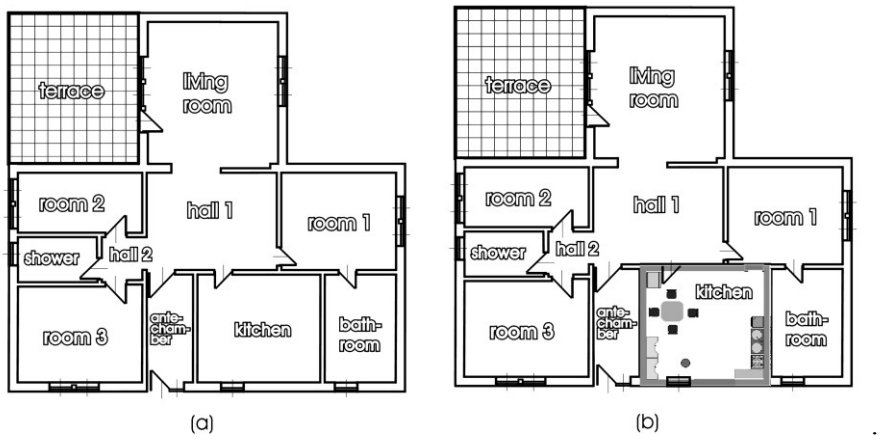
Rys. 5.2. Wybrane reguły gramatyki hipergrafowej generującej struktury rozkładów pomieszczeń
Fig. 5.2. The chosen rules of the hypergraph grammar generating structures of floor layouts

Czwarta produkcja dodaje salon połączony z tarasem. Produkcje *p*5 i *p*6 umożliwiają otrzymanie różnych konfiguracji pomieszczeń złożonych z dwóch pokoi, prysznica i małego holu. Pierwsza z tych produkcji umieszcza prysznic pomiędzy pokojami, podczas gdy druga umieszcza prysznic w taki sposób, aby przylegał on tylko do jednego z pokoi, ale miał dwa wejścia, jedno z holu i jedno z pokoju. Hipergraf wygenerowany z wykorzystaniem tej gramatyki jest pokazany na rys. 5.3, a jeden z możliwych rozkładów pomieszczeń odpowiadających temu hipergrafowi na rys. 5.4a. W diagramie z rys. 5.4a relacja dostępności jest przedstawiona za pomocą ikon reprezentujących drzwi, a na ścianach zostały umieszczone ikony

reprezentujące okna, gdyż będzie to miało istotne znaczenie w dalszych rozważaniach, dotyczących aranżacji wnętrz i ustawiania mebli w poszczególnych pomieszczeniach. ◊



Rys. 5.3. Hipergraf wygenerowany gramatyką hipergrafową z rys. 5.2
Fig. 5.3. A hypergraph generated by the hypergraph grammar from fig. 5.2



Rys. 5.4. Rozkłady pomieszczeń: a) rozkład odpowiadający hipergrafowi z rys.5.3, b) ten sam rozkład z umieszczonymi w kuchni sprzętami
Fig. 5.4. Floor layouts: a) a layout corresponding to the hypergraph from fig. 5.3, b) the same layout with the kitchen arrangement

Produkcja p może być zastosowana do hierarchicznego hipergrafu H, jeśli jest spełniony jej predykat stosowalności. Zastosowanie produkcji bezkontestowej hierarchicznej gramatyki hipergrafowej $p = (l, r, \xi, sr)$ polega na zastąpieniu w H hipergrafu izomorficznego z l z dokładnością do funkcji zagnieżdżenia hipergrafem r, zastąpieniu zewnętrznych wierzchołków usuwanego hipergrafu izomorficznych z wierzchołkami z ext_l odpowiednimi wierzchołkami zewnętrznymi z ext_r i określeniu wartości atrybutów przypisanych elementom z r zgodnie z regułami semantycznymi z sr. Jeśli hiperkrawędź izomorficzna z hiperkrawędzią z l posiada w H potomków wyznaczonych przez funkcję ch_H, to elementy te zostają zagnieżdżone w jednej z hiperkrawędzi obiektowych r o takiej samej etykiecie, jaką posiada hiperkrawędź z l.

Definicja 5.3:

Niech $\mathcal{G} = (V, E, P, X)$ będzie bezkontekstową hierarchiczną hipergrafową gramatyką rozmieszczenia, a H i H' będą dwoma atrybutowanymi hierarchicznymi hipergrafami.

H' jest **bezpośrednio wyprowadzone** z H ($H \Rightarrow H'$), jeśli istnieje produkcja $p = (l, r, \xi, sr)$ w $\mathcal{G}$ taka, że:

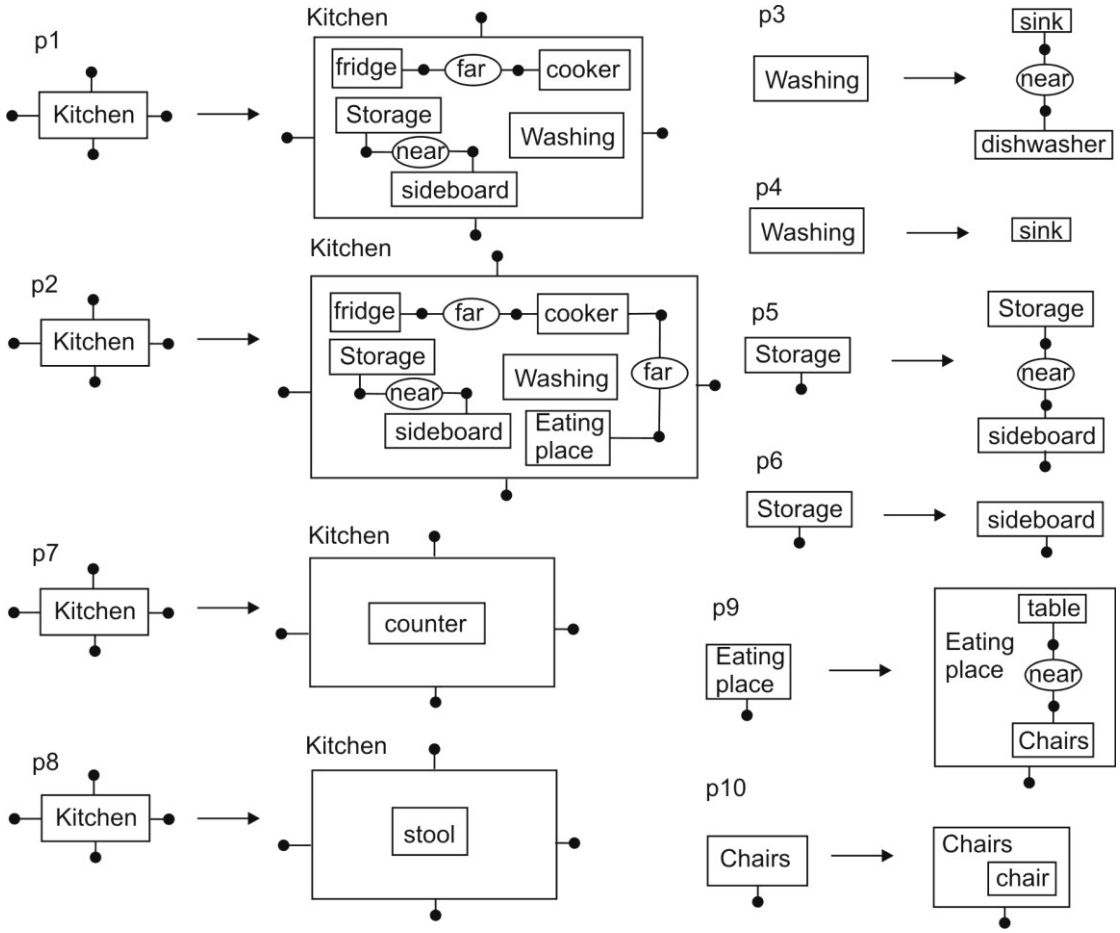
- ξ jest spełnione dla H,
- h jest podgrafem H izomorficznym z l z dokładnością do funkcji zagnieżdżenia poprzez izomorfizm ρ, tzn. $ch_H(\rho(e))$ dla hiperkrawędzi e należącej do l może nie być zbiorem pustym,
- H' jest izomorficzne z wynikiem zastąpienia h w H przez r, zagnieżdżenia elementów $ch_H(\rho(e))$ w hiperkrawędzi obiektowej e' hipergrafu r takiej, że $lb_r(e') = lb_H(\rho(e))$, zastąpienia zewnętrznych wierzchołków h odpowiednimi zewnętrznymi wierzchołkami r i przypisania wartości atrybutom elementów z r zgodnie z regułami ze zbioru sr. □

Przykład 5.3: aranżacja pomieszczeń

Kilka produkcji bezkontekstowej hierarchicznej hipergrafowej gramatyki rozmieszczenia generującej struktury opisujące rozkład sprzętów w kuchni jest przedstawionych na rys. 5.5. Predykaty stosowalności produkcji i reguły semantyczne zostały pominięte dla uproszczenia prezentacji. Dwie pierwsze produkcje umożliwiają stworzenie rozkładu kuchni z kącikiem jadalnym lub bez niego. Produkcja generująca hiperkrawędź reprezentującą kącik jadalny jest wykorzystywana tylko wtedy gdy wartość atrybutu *area* przypisanego hiperkrawędzi *Kitchen* (reguła $p2$) jest większa niż 10 m^2. W przeciwnym przypadku wszystkie sprzęty nie zmieściłyby się lub byłyby ustawione zbyt ciasno. Trzecia i czwarta produkcja umożliwiają umieszczenie w kuchni odpowiednio zlewu i zmywarki do naczyń lub samego zlewu. Następne dwie produkcje pozwalają dodać wymaganą liczbę szafek, które powinny się znajdować obok siebie. Produkcje $p7$ i $p8$ umożliwiają dodanie odpowiednio lady i taboretu. Ostatnie dwie produkcje pozwalają umieścić stół z kilkoma krzesłami w kąciku jadalnym.

Należy zauważyć, że na przykład zastosowanie produkcji $p7$ do generowanego hipergrafu spowoduje dodanie nowej hiperkrawędzi obiektowej reprezentującej ladę, która będzie zagnieżdżona w hiperkrawędzi reprezentującej kuchnię. Atomy generowanego hipergrafu, które były zagnieżdżone w hiperkrawędzi reprezentującej kuchnię, pozostaną zagnieżdżone w hiperkrawędzi etykietowanej *Kitchen*. ◊

Język generowany przez daną bezkontekstową hierarchiczną hipergrafową gramatykę rozmieszczenia jest zbiorem hierarchicznych hipergrafów wygenerowanych z aksjomatu gramatyki.



Rys. 5.5. Bezkontekstowa hierarchiczna hipergrafowa gramatyka rozmieszczenia generująca struktury rozkładów kuchni

Fig. 5.5. A context-free hierarchical hypergraph layout grammar generating structures of kitchen arrangements

Definicja 5.4:

Niech $\mathcal{G} = (V, E, P, X)$ będzie bezkontekstową hierarchiczną hipergrafową gramatyką rozmieszczenia.

Język generowany przez $\mathcal{G}$ jest zbiorem $L(\mathcal{G}) = \{H \in \mathcal{H} \mid X \Rightarrow^* H\}$. □

5.2. Programowane hierarchiczne gramatyki hipergrafowe

W celu zwiększenia mocy generacyjnej bezkontekstowych hierarchicznych hipergrafowych gramatyk rozmieszczenia wyposażamy je w diagramy sterujące, otrzymując programowane hierarchiczne gramatyki hipergrafowe. Diagram sterujący określający porządek, w jakim mogą być stosowane produkcje gramatyki, jest skierowanym grafem, którego wierzchołki etykietowane są nazwami produkcji. Ponadto, istnieją dwa wyróżnione wierzchołki,

początkowy – etykietowany przez I, do którego nie wchodzą żadne krawędzie, i końcowy – etykietowany przez F, z którego nie wychodzą żadne krawędzie.

Definicja 5.5:

Niech P będzie zbiorem produkcji bezkontekstowej hierarchicznej hipergrafowej gramatyki rozmieszczenia $\mathcal{G}$. Niech $\Omega = P \cup \{I, F\}$ będzie zbiorem etykiet wierzchołkowych.

Diagramem sterującym nad Ω zdefiniowanym dla $\mathcal{G}$ nazywamy skierowany, etykietowany wierzchołkowo graf $CD = (V, E, lb_V)$, gdzie:

- V – skończony zbiór wierzchołków,
- $E \subseteq V \times V$ – zbiór skierowanych krawędzi,
- lb_V: $V \to \Omega$ – funkcja etykietowania wierzchołków,
- istnieje dokładnie jeden wierzchołek etykietowany przez I i dokładnie jeden etykietowany przez F,
- nie istnieją krawędzie wchodzące do wierzchołka etykietowanego przez I ani krawędzie wychodzące z wierzchołka etykietowanego przez F. □

Definicja 5.6:

Programowana hierarchiczna hipergrafowa gramatyka rozmieszczenia jest parą $\mathcal{G}_P = (\mathcal{G}, CD)$, gdzie:

- $\mathcal{G}$ jest bezkontekstową hierarchiczną hipergrafową gramatyką rozmieszczenia,
- CD jest diagramem sterującym zdefiniowanym dla $\mathcal{G}$. □

Diagram sterujący określa kolejność, w jakiej powinny być stosowane produkcje gramatyki. Jeśli w generowanym hipergrafie nie istnieje podgraf izomorficzny z lewą stroną produkcji, która powinna być w danym momencie wykorzystana, to jest stosowana inna produkcja wyznaczona przez diagram lub generacja jest przerywana. Język generowany przez programowaną hierarchiczną hipergrafową gramatykę rozmieszczenia jest zbiorem hierarchicznych hipergrafów, które da się wyprowadzić w gramatyce, przechodząc od początkowego do końcowego wierzchołka diagramu sterującego.

Definicja 5.7:

Niech $\mathcal{G}_P = (\mathcal{G}, CD)$ będzie programowaną hierarchiczną hipergrafową gramatyką rozmieszczenia. Niech H_1 i H_2 będą dwoma atrybutowanymi hierarchicznymi hipergrafami rozmieszczenia, a p produkcją gramatyki $\mathcal{G}$. Niech v_1 i v_2 będą dwoma wierzchołkami CD.

Para (H_1, v_1) **bezpośrednio wyprowadza** parę (H_2, v_2) $((H_1, v_1) \Rightarrow (H_2, v_2))$, jeśli:

- $H_1 \Rightarrow H_2$ za pomocą produkcji p i istnieje w CD krawędź (v_1, v_2), gdzie $lb_V(v_2) = p$, lub

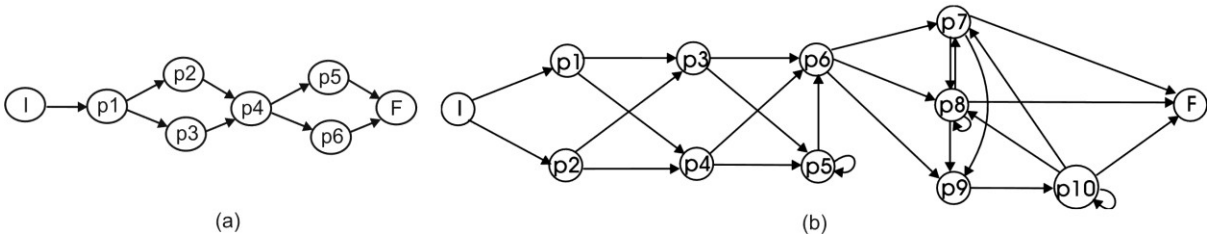
- $X \Rightarrow H_2$ za pomocą produkcji p, gdzie $X = H_1$ jest aksjomatem $\mathcal{G}$ i istnieje w CD krawędź (v_1, v_2), gdzie $lb_V(v_1) = I$ i $lb_V(v_2) = p$, lub
- $H_1 = H_2$ i istnieje w CD krawędź (v_1, v_2), gdzie $lb_V(v_2) = F$. □

Definicja 5.8:
Niech $\mathcal{G}_P = (\mathcal{G}, CD)$ będzie programowaną hierarchiczną hipergrafową gramatyką rozmieszczenia i niech v_I oraz v_F oznaczają początkowy i końcowy wierzchołek CD.

Język generowany przez $\mathcal{G}_P$ jest zbiorem $L(\mathcal{G}_P) = \{H \in L(\mathcal{G}) \mid (X, v_I) \Rightarrow^* (H, v_F)\}$. □

Przykład 5.4: rozkłady pomieszczeń
Diagram sterujący dla gramatyki hipergrafowej z rys. 5.2 jest przedstawiony na rys. 5.6a. Każda ścieżka od wierzchołka początkowego I do wierzchołka końcowego F reprezentuje jeden możliwy sposób otrzymania hipergrafu reprezentującego potencjalny rozkład pomieszczeń. Zastosowanie sekwencji produkcji $p1, p2, p4, p6$ daje w wyniku hipergraf z rys. 5.3.

Diagram sterujący dla gramatyki hipergrafowej z rys.5.5 jest pokazany na rys. 5.6b. Zastosowanie ciągu produkcji $p2, p3, p6, p7, p8, p9$ i $4 \times p10$ prowadzi do hipergrafu z rys. 3.5a. ◊



Rys. 5.6. Diagramy sterujące: a) dla gramatyki hipergrafowej generującej struktury rozkładów pomieszczeń, b) dla gramatyki hipergrafowej generującej struktury rozkładów kuchni
Fig. 5.6. Control diagrams: a) for the hypergraph grammar generating floor layout structures, b) for the hypergraph grammar generating kitchen arrangement structures

Należy zauważyć, że pomimo iż diagram sterujący dla gramatyki hipergrafowej wyznacza kolejność stosowania reguł gramatyki, to nadal istnieje możliwość wygenerowania wielu hipergrafów odpowiadających różnorodnym projektom, gdyż w ogólnym przypadku można zwykle wyspecyfikować w diagramie znaczną lub nawet nieskończoną liczbę ścieżek prowadzących do różnych rozwiązań.

5.3. Podsumowanie

W rozdziale tym zostały przedstawione narzędzia służące do generowania wewnętrznych reprezentacji potencjalnych rozwiązań projektowych. Została zdefiniowana hierarchiczna hi-

pergrafowa gramatyka rozmieszczenia, generująca hierarchiczne hipergrafy reprezentujące struktury diagramów projektowych, oraz jej postać bezkontekstowa. Zostało omówione zastosowanie produkcji bezkontekstowej hipergrafowej gramatyki rozmieszczenia będące lokalną transformacją wyprowadzanego hipergrafu zależną od warunków określonych predykatem stosowalności produkcji. Został również określony język generowany za pomocą gramatyki tego typu. Następnie została zdefiniowana programowana hierarchiczna hipergrafowa gramatyka rozmieszczenia, która jest wyposażona w diagram sterujący określający porządek, w jakim mogą być stosowane produkcje gramatyki. Mechanizm ten służy do kontrolowania sposobu przeszukiwania przestrzeni dopuszczalnych rozwiązań danego problemu projektowego. Wprowadzone pojęcia zostały zilustrowane przykładami hierarchicznych hipergrafowych gramatyk rozmieszczenia generujących hierarchiczne hipergrafy, należące do rodziny struktur reprezentujących rozkłady i aranżacje pomieszczeń.

Wykorzystanie hierarchicznych hipergrafowych gramatyk rozmieszczenia przez agentów będących inteligentnymi asystentami wspomagającymi proces projektowy zostanie omówione w rozdziale 7.

6. WNIOSKOWANIE Z HIPERGRAFOWEJ REPREZENTACJI DIAGRAMÓW

W rozdziale tym zostanie omówiony logiczny model wnioskowania o projektach z wykorzystaniem wewnętrznej reprezentacji diagramów projektowych. Diagramy te są opisywane za pomocą formuł logicznych, których syntaktyka bazuje na wielorodzajowej sygnaturze. Ontologiczne dopasowanie symboli sygnatury do pojęć konceptualizacji projektowej umożliwia wykorzystanie logiki pierwszego rzędu do wyrażania wiedzy dotyczącej obiektów projektowych reprezentowanych przez komponenty diagramów. Semantyka formuł logicznych jest określana za pomocą interpretacji definiowanej poprzez strukturę relacyjną przypisującą atomy hipergrafów i wartości ich atrybutów elementom składowym formuł. Wiedza o projektach przechowywana w hipergrafowej reprezentacji diagramów pozwala więc ocenić zgodność generowanych rozwiązań, zarówno ukończonych, jak i częściowych, z zadanymi kryteriami projektowymi [Grab10, GrŚl11].

Wizualny język projektowania scharakteryzowany w rozdziale 2 umożliwia projektantowi śledzenie zmian zachodzących w diagramach projektowych. Każdy uogólniony diagram, będący elementem języka *DJP*, reprezentuje określony etap rozwoju rozwiązania projektowego i niesie informację umożliwiającą wnioskowanie syntaktyczne i semantyczne. System wspomagający konceptualną fazę projektowania pomaga użytkownikowi podczas procesu projektowego poprzez wnioskowanie z wewnętrznej reprezentacji diagramów będących elementami języka wizualnego. Wiedza zawarta w tej reprezentacji jest transformowana do postaci formuł logicznych opisujących generowane rozwiązania. Moduł wnioskowania sprawdza, czy hierarchiczne hipergrafy rozmieszczenia spełniają określone warunki poprzez porównanie otrzymanych formuł z formułami logicznymi reprezentującymi normy i ograniczenia projektowe oraz wprowadzone wcześniej wymagania projektowe [GrŚL08]. Jeśli rozwiązanie nie jest zgodne z określonymi kryteriami projektowymi, system sugeruje projektantowi niezbędne modyfikacje.

6.1. Syntaktyka formuł logicznych opisujących diagramy projektowe

W celu formalnej charakteryzacji wnioskowania dotyczącego diagramów zdefiniujemy zbiór formuł logicznych pierwszego rzędu opisujących własności diagramów projektowych. Dla ujednolicenia reprezentacji wiedzy zawartej w diagramach zostanie wykorzystany wielorodzajowy język logiczny, którego formuły są tworzone z symboli wielorodzajowej sygnatury [LiMP08] zwanej słownikiem.

Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową.

Niech $D = \bigcup_{a \in A} D_a$ oznacza zbiór wartości atrybutów ze zbioru A.

Definicja 6.1:
Wielorodzajowym słownikiem formuł logicznych dla diagramów projektowych nazywamy trójkę $\mathcal{V} = \{\mathcal{S}, \mathcal{F}, \mathcal{P}\}$, gdzie:

- $\mathcal{S} = \{O, D\}$ jest zbiorem rodzajów,
- $\mathcal{F}$ jest zbiorem symboli n-arnych funkcji, z których każda jest typu $s_1 \times \ldots \times s_n \rightarrow s$, gdzie $n \geq 0$, $s_i, s \in \mathcal{S}$, przy czym funkcje 0-argumentowe są symbolami stałych,
- $\mathcal{P}$ jest zbiorem symboli n-arnych predykatów o argumentach postaci $s_1 \times \ldots \times s_n$, gdzie $n \geq 0$, $s_i \in \mathcal{S}$. □

Aby formuły logiczne, których syntaktyka będzie się składać z symboli występujących w słowniku $\mathcal{V}$, opisywały diagramy projektowe, symbole te powinny być interpretowane zgodnie ze zdefiniowaną wcześniej konceptualizacją projektową $\mathcal{C}$. Odpowiednią intensjonalną interpretację symboli słownika zapewnia odwzorowanie zwane dopasowaniem ontologicznym [LiMP08] między $\mathcal{V}$ i $\mathcal{C}$.

Definicja 6.2:
Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ będzie konceptualizacją projektową, a $\mathcal{V} = \{\mathcal{S}, \mathcal{F}, \mathcal{P}\}$ – słownikiem formuł logicznych dla diagramów projektowych.

Dopasowaniem ontologicznym między $\mathcal{V}$ i $\mathcal{C}$ nazywamy częściowe odwzorowanie $\mathcal{I}: \mathcal{V} \rightarrow \mathcal{C}$, które spełnia następujące warunki:

- symbolom stałych rodzaju O przypisuje obiekty aktywne ze zbioru $O_R \subset O$,
- symbolom funkcji o argumentach rodzaju O i wartościach rodzaju D przypisuje atrybuty ze zbioru A,
- symbolom jednoargumentowych predykatów o argumentach rodzaju O przypisuje typy obiektów ze zbioru Z,
- symbolom wieloargumentowych predykatów o argumentach rodzaju O przypisuje relacje ze zbioru R. □

Podstawowe fakty dotyczące diagramów mają postać $P(c_1,\ldots, c_n)$, gdzie P jest symbolem n-arnej relacji, a $c_1,\ldots, c_n$ są symbolami stałych rodzaju O.

Przykład 6.1: rozkłady pomieszczeń
Elementami słownika formuł logicznych w przypadku projektowania rozkładów pomieszczeń są:

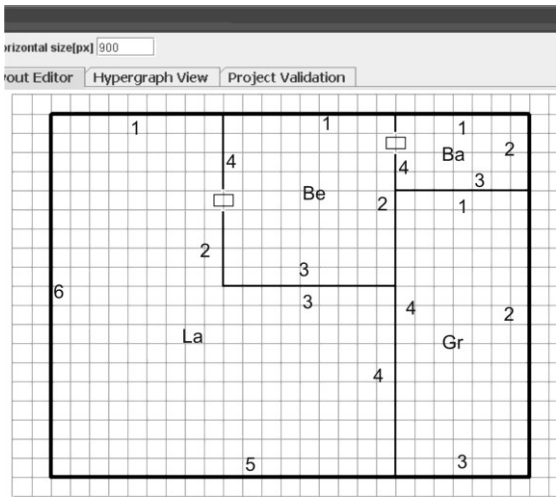
1. zbiór rodzajów $\mathcal{S}$ złożony ze zbioru O obiektów reprezentujących pomieszczenia, obszary i ściany, oraz zbioru D będącego sumą zbioru liczb rzeczywistych $\mathcal{R}$ i zbioru kierunków geograficznych określających orientację ścian,
2. zbiór symboli funkcji $\mathcal{F}$, który zawiera:
   - symbole stałych rodzaju O odpowiadające obiektom aktywnym reprezentującym ściany, i symbole stałych rodzaju D,
   - symbol *blg*(O): O funkcji, która określa przynależność obiektów aktywnych do obiektów podstawowych (identycznościowa dla obiektów podstawowych),
   - symbole *area*, *length*, *order*, *window_number*, *door_number*(O): $\mathcal{R}$, *position*(O): $\mathcal{R}^2$ i *orientation*(O): {*N*, *S*, *W*, *E*} funkcji częściowych, które odpowiadają atrybutom obiektów projektowych,
   - symbole funkcji arytmetycznych: +, –, *, /($\mathcal{R}^2$): $\mathcal{R}$,
3. zbiór symboli predykatów $\mathcal{P}$, który zawiera:
   - symbole *room*, *zone*, *wall*, *wall_pass*(O) predykatów określających typy obiektów projektowych,
   - symbole *acc*, *adj*, *cnts*(O^n), $n > 1$, predykatów określających relacje zachodzące między obiektami projektowymi,
   - symbole predykatów relacyjnych: <, >($\mathcal{R}^2$).

Dla diagramu przedstawionego na rys. 6.1 podstawowe fakty opisujące relacje zachodzące pomiędzy komponentami rozkładu pomieszczeń dotyczą dostępności pomiędzy pomieszczeniami: *acc*(*La*.2, *Be*.4), *acc*(*Be*.2, *Ba*.4) oraz przyległości pomiędzy pomieszczeniami: *adj*(*La*.3, *Be*.3), *adj*(*La*.4, *Gr*.4), *adj*(*Be*.2, *Gr*.4), *adj*(*Ba*.3, *Gr*.1). ◊

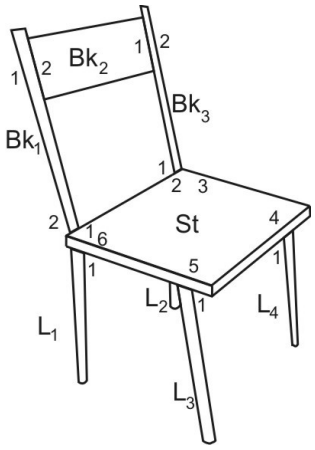
Przykład 6.2: krzesła
Słownik formuł logicznych w przypadku projektowania krzeseł zawiera zbiór rodzajów $\mathcal{S}$ złożony ze zbioru O obiektów reprezentujących siedzisko, oparcie, podstawę, oraz elementy oparcia, poręczy, nogi krzeseł i punkty ich połączenia, oraz zbioru D będącego sumą zbioru liczb rzeczywistych $\mathcal{R}$ i zbiorów kolorów, rodzajów materiałów i tekstur. Zbiór symboli funkcji $\mathcal{F}$ zawiera symbole stałych rodzaju D i O, gdzie stałe rodzaju O odpowiadają obiektom aktywnym reprezentującym punkty połączeń elementów krzesła, symbol funkcji *blg*(O): O, która określa przynależność obiektów aktywnych do obiektów podstawowych,

symbole jednoargumentowych funkcji częściowych *position*, *order*, *width*, *height*, *depth*, *colour*, *material*, *texture* o argumencie rodzaju *O*, które odpowiadają atrybutom obiektów projektowych, oraz symbole funkcji arytmetycznych. Zbiór symboli predykatów $\mathcal{P}$ zawiera jednoargumentowe symbole *seat*, *back*, *back_element*, *arm_element*, *base*, *leg*, *connection_point* określające typy obiektów projektowych, symbole *adj*, *cnts,* określające relacje zachodzące między obiektami projektowymi, oraz symbole <, >.

Dla diagramów języka opisującego krzesła relacja *adj* jest wyznaczona przez stykające się ze sobą elementy krzeseł. Podstawowe fakty zachodzące dla diagramu krzesła przedstawionego na rys. 6.2 są następujące: $adj(Bk_1.2, St.1)$, $adj(Bk_1.1, Bk_2.2)$, $adj(Bk_2.1, Bk_3.2)$, $adj(Bk_3.1, St.2)$, $adj(L_1.1, St.6)$, $adj(L_2.1, St.5)$, $adj(L_3.1, St.4)$, $adj(L_4.1, St.3)$. ◊



Rys. 6.1. Przykładowy diagram rozkładu pomieszczeń w programie *HSSDR*
Fig. 6.1. An example of a floor layout diagram in *HSSDR* program



Rys. 6.2. Przykładowy diagram krzesła
Fig. 6.2. An example of a chair diagram

Ponieważ rozważane tu formuły logiczne służą do opisu i wnioskowania dotyczącego własności obiektów projektowych, będziemy rozważać tylko zmienne rodzaju *O*. Zakładamy istnienie nieskończonego zbioru zmiennych tego rodzaju oznaczanych jako *x* i *y* z odpowied-

nimi indeksami. Zbiór termów T jest tworzony przez domknięcie symboli stałych i zmiennych nad zastosowaniem funkcji, tzn. jeśli f jest symbolem n-arnej funkcji, a $t_1,\ldots, t_n$ są termami rodzaju zgodnego z rodzajem argumentów funkcji f, wówczas $f(t_1,\ldots, t_n)$ także jest termem.

Zbiór $\mathcal{B}$ atomowych formuł logicznych nad słownikiem $\mathcal{V}$ składa się z formuł postaci:

- $P(t_1,\ldots, t_n)$, gdzie P jest symbolem n-arnego predykatu, a $t_1,\ldots, t_n$ są termami takiego samego rodzaju jak rodzaje argumentów predykatu P,
- $t_1 = t_2$, gdzie t_1 i t_2 są termami tego samego rodzaju.

Zbiór formuł logicznych zbudowanych z formuł atomowych określonych nad słownikiem $\mathcal{V}$ z wykorzystaniem negacji, spójników logicznych i kwantyfikatorów, który jest spójny i domknięty nad relacją wynikania, będzie oznaczany przez $\mathcal{F}$. Spójność zbioru $\mathcal{F}$ oznacza, że formuły nie dostarczają informacji sprzecznych. Formuły z $\mathcal{F}$ zawierają zmienne uniwersalnie kwantyfikowane nad odpowiednimi typami obiektów. *Formuły* zbioru $\mathcal{F}$ definiujemy nad zbiorem $\mathcal{B}$ i zbiorem zmiennych indukcyjnie:

- jeśli $\phi \in \mathcal{B}$, to ϕ jest formułą,
- jeśli ϕ i φ są formułami, to $\neg\phi$, $\phi \wedge \varphi$ też są formułami,
- jeśli ϕ jest formułą, a x jest zmienną, to $\exists\, x\phi$ też jest formułą.

Formuły niezawierające zmiennych wolnych są zdaniami opisującymi diagramy projektowe. Elementy z $\mathcal{F}$ charakteryzują więc ogólne własności diagramów projektowych, będących elementami języka *DJP* i stanowią logiczne konsekwencje podstawowych informacji zawartych w projektach.

Przykładowe zdania ze zbioru $\mathcal{F}$ dla języka opisującego rozkłady pomieszczeń są następujące:

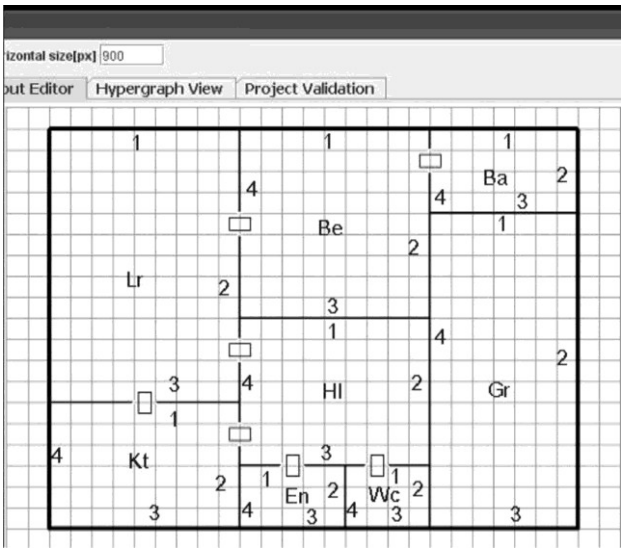
- $\forall\ t \in room\ \ area(t) > 10$ – powierzchnia wszystkich pomieszczeń jest większa niż 10 m^2,
- $(\exists\, x \in wall\colon blg(x) = Kt \wedge window\_number(x) = 1) \wedge (\exists\, y \in wall\_pass\colon y \neq x \wedge blg(y) = Kt \wedge door\_number(y) = 2)$ – kuchnia ma co najmniej jedno okno i dwoje drzwi,
- $\exists\, x_1,\ldots, x_n, y_1,\ldots, y_n,\quad x \in wall\_pass\colon blg(x_i) = Hl,\quad i = 1,\ldots, n,\quad blg(y_i) \neq blg(y_j) \neq Hl,$ $i, j = 1,\ldots, n,\ i \neq j,\ acc(x_1, y_1) \wedge\ldots\wedge acc(x_n, y_n)$ – istnieje n pomieszczeń dostępnych z holu,
- $\forall\, x_1,\ldots, x_n \in wall\_pass\colon blg(x_i) = blg(x_{i+1}),\quad i = 2, 4,\ldots, n\text{-}2,\quad n \geq 4,\quad acc(x_1, x_2) \wedge\ldots\wedge acc(x_{n-1}, x_n) \Rightarrow acc(x_1,\ldots, x_n)$ – pomieszczenie, do którego należy ściana x_n, jest pośrednio dostępne z pomieszczenia, do którego należy ściana x_1,

- $\forall\ t_1, t_2, t_3 \in \{zone, room\}\ cnts(t_2, t_1) \wedge cnts(t_3, t_2) \Rightarrow cnts(t_3, t_1)$ – przechodniość zawierania się pomieszczeń i obszarów.

Przykład 6.3: rozkłady pomieszczeń
Dla diagramu rozkładu pomieszczeń przedstawionego na rys. 6.3 następujące zdania są prawdziwe:

1. $acc(Lr.2, Be.4) \wedge acc(Be.2, Ba.4) \Rightarrow acc(Lr.2, Be.4, Be.2, Ba.4)$,
2. $acc(Hl.4, Kt.2) \wedge acc(Hl.4, Lr.2) \wedge acc(Hl.3, Wc.1) \wedge acc(Hl.4, En.1)$.

Pierwsze z nich oznacza, że łazienka jest dostępna z części rekreacyjnej poprzez sypialnię, drugie natomiast, że istnieją cztery pomieszczenia dostępne z holu. ◊



Rys. 6.3. Diagram projektowy rozkładu pomieszczeń w programie *HSSDR*
Fig. 6.3. A floor layout design diagram in *HSSDR* program

6.2. Semantyka formuł logicznych opisujących diagramy projektowe

Semantyka jest nadawana formułom ze zbioru $\mathcal{F}$ za pomocą ekstensjonalnej interpretacji, która przypisuje symbolom ze słownika $\mathcal{V}$ elementy zdefiniowane nad pewną dziedziną. Interpretacja ta jest tutaj definiowana poprzez strukturę relacyjną, która składa się z dziedziny określonych elementów i odpowiednich przypisań [FHMV95].

Definicja 6.3:
$\mathcal{V}$- struktura relacyjna *J* składa się z:

- dziedziny $dom(J) = dom(J_O) \cup dom(J_D)$ zawierającej dziedziny odpowiednio dla symboli rodzaju O i rodzaju D,
- przypisania elementu $c^J \in dom(J_O)$ każdemu symbolowi stałej c rodzaju O,
- przypisania elementu $c^J \in dom(J_D)$ każdemu symbolowi stałej c rodzaju D,

- przypisania n-arnego predykatu $P^J \subseteq dom(J)^n$ każdemu n-arnemu symbolowi predykatu $P \in \mathcal{P}$ z zachowaniem rodzaju argumentów,
- przypisania n-arnej funkcji $f^J: dom(J)^n \rightarrow dom(J)$ każdemu n-arnemu symbolowi funkcji $f \in \mathcal{F}$ z zachowaniem rodzaju argumentów i wartości. □

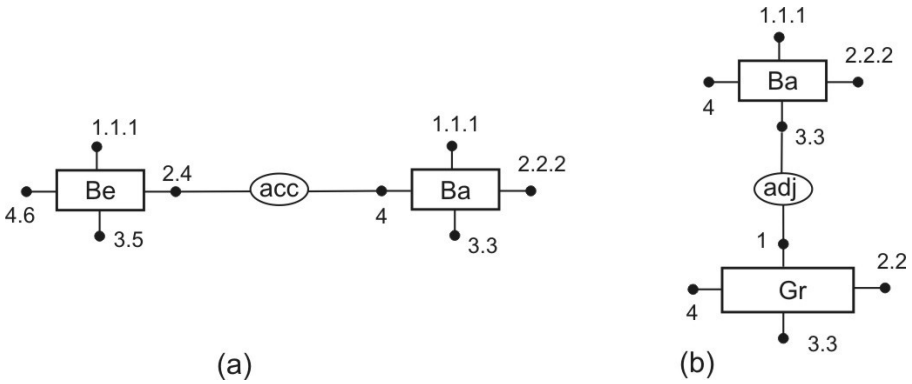
Niech $\mathcal{H}$ oznacza rodzinę atrybutowanych hierarchicznych hipergrafów rozmieszczenia ze zbiorem atomów $\mathcal{AT} = \mathcal{V} \cup \mathcal{E}$ i rodziną Ev funkcji częściowych określających wartości atrybutów hiperkrawędzi obiektowych i wierzchołków hipergrafów z $\mathcal{H}$. Ponieważ projektowane artefakty są reprezentowane za pomocą atrybutowanych hierarchicznych hipergrafów rozmieszczenia, dziedzinę struktury relacyjnej $dom(J_O)$ dla elementów rodzaju O stanowi zbiór $\mathcal{V} \cup \mathcal{E}_C$ wierzchołków i hiperkrawędzi obiektowych tych hipergrafów. Dziedzinę $dom(J_D)$ dla elementów rodzaju D stanowi zbiór wartości odwzorowania Ev.

Struktura relacyjna J przypisuje symbolom stałych rodzaju O wierzchołki hipergrafu reprezentujące obiekty aktywne. Interpretacją symbolu jednoargumentowej funkcji *blg* o argumencie i wartości rodzaju O jest funkcja blg^J przypisująca hiperkrawędź obiektową jednemu z jej wierzchołków docelowych lub samej tej hiperkrawędzi. Interpretacją symboli funkcji o argumentach rodzaju O i wartościach rodzaju D są atrybuty atomów hipergrafów. Interpretacją symboli predykatów o argumentach rodzaju O są hiperkrawędzie relacyjne reprezentujące relacje zachodzące w hipergrafie. Relacje w strukturach projektów są specyfikowane pomiędzy obiektami aktywnymi będącymi fragmentami innych obiektów i odpowiadającymi wierzchołkom hipergrafów. Interpretacją symbolu $n+m$-arnego predykatu $P(s_1,\ldots, s_{n+m})$, gdzie $s_i \in O$, jest hiperkrawędziowa relacja P^J w hipergrafie taka, że istnieje hiperkrawędź relacyjna e wychodząca z sekwencji zewnętrznych wierzchołków $v_1,..., v_n$ co najmniej jednego uchwytu hipergrafu i wchodząca do sekwencji zewnętrznych wierzchołków $w_1,..., w_m$ należących do innych uchwytów i taka, że $s(e) = v_1,..., v_n$ i $t(e) = w_1,..., w_m$, gdzie s i t są odwzorowaniami wyznaczającymi sekwencje wierzchołków źródłowych i docelowych dla hiperkrawędzi danego hipergrafu. Interpretacją symboli predykatów o argumentach rodzaju D są ograniczenia wiążące atrybuty przypisane atomom hipergrafów.

Podstawowy fakt postaci $P(c_1,\ldots, c_n)$, gdzie $c_1,\ldots, c_n$ są symbolami stałych rodzaju O, zachodzi dla danego diagramu, jeśli w hipergrafie reprezentującym ten diagram istnieje odpowiednia hiperkrawędź relacyjna etykietowana P^J, której wierzchołki źródłowe i docelowe odpowiadające symbolom stałych należą do uchwytów reprezentujących komponenty diagramu. Hiperkrawędź relacyjna wraz z uchwytami, których wierzchołki są przez nią połączone, tworzy podgraf hipergrafu reprezentującego strukturę projektu.

Przykład 6.4: rozkłady pomieszczeń

Fakt *acc*(*Be*.2, *Ba*.4) opisujący relację dostępności pomiędzy sypialnią i łazienką poprzez ich wspólną ścianę (rys. 6.1) jest interpretowany jako podgraf hipergrafu z rys. 4.3 pokazany na rys. 6.4a, a fakt *adj*(*Ba*.3, *Gr*.1) opisujący relację przyległości pomiędzy łazienką i garażem jest interpretowany jako podgraf tego samego hipergrafu przedstawiony na rys. 6.4b. Pierwsza część atrybutu *order* każdego wierzchołka odpowiada numerowi ściany ograniczającej obszar lub pomieszczenie. ◊



Rys. 6.4. Podgrafy: a) reprezentujący dostępność pomiędzy łazienką i sypialnią, b) reprezentujący przyległość łazienki i garażu

Fig. 6.4. Subgraphs: a) representing accessibility between the bathroom and bedroom, b) representing adjacency between the bathroom and garage

Kolejnym krokiem w definiowaniu semantyki formuł pierwszego rzędu jest specyfikacja interpretacji zmiennych [FHMV95]. Wartościowanie na strukturze relacyjnej J jest funkcją odwzorowującą zmienne w elementy dziedziny $dom(J)$.

Definicja 6.4:

Niech N oznacza zbiór zmiennych występujących w formułach z $\mathcal{F}$. Niech J będzie $\mathcal{V}$- strukturą relacyjną.

Funkcja $\omega: N \rightarrow V \subset dom(J_O)$, która przypisuje wierzchołki hipergrafów zmiennym z N, będzie nazywana **wartościowaniem zmiennych** na J. □

Dla danej $\mathcal{V}$- struktury relacyjnej J i wartościowania zmiennych ω na J, ω jest indukcyjnie rozszerzane do funkcji odwzorowującej termy w elementy dziedziny $dom(J)$. Niech $\omega(c) = c^J$ dla każdego symbolu stałej c, gdzie $c^J \in dom(J_O)$, jeśli c jest rodzaju O i $c^J \in dom(J_D)$, jeśli c jest rodzaju D. Wówczas definicja funkcji ω jest rozszerzana przez indukcję na strukturę termów w taki sposób, że $\omega(f(t_1,\ldots, t_n)) = f^J(\omega(t_1),\ldots, \omega(t_n))$.

Niech ω będzie wartościowaniem na danej strukturze relacyjnej J. $(J, \omega) \models \phi$ oznacza, że formuła $\phi \in \mathcal{F}$ jest spełniona w J przy wartościowaniu ω. Spełnialność formuł ze zbioru $\mathcal{F}$ jest określona następująco:

- $(J, \omega) \models {}_P(t_1,..., t_n)$, gdzie ${}_P \in \mathcal{P}$ jest symbolem n-arnego predykatu, a $t_1,\ldots, t_n$ są termami takiego samego rodzaju jak rodzaje argumentów predykatu ${}_P \Leftrightarrow (\omega(t_1),..., \omega(t_n)) \in {}_P^J$,
- $(J, \omega) \models t_1 = t_2$, gdzie t_1 i t_2 są termami tego samego rodzaju $\Leftrightarrow \omega(t_1) = \omega(t_2)$,
- $(J, \omega) \models \neg\phi \Leftrightarrow (J, \omega) \not\models \phi$,
- $(J, \omega) \models \phi_1 \wedge \phi_2 \Leftrightarrow (J, \omega) \models \phi_1 \wedge (J, \omega) \models \phi_2$,
- $(J, \omega) \models \exists\, x\phi \Leftrightarrow (J, \omega[x/v]) \models \phi$ dla $v \in dom(J_O)$, gdzie $\omega[x/v]$ oznacza wartościowanie takie, że $\omega(x) = v$.

Atomowa formuła postaci ${}_P(t_1,..., t_n)$ o argumentach rodzaju O zachodzi dla danego diagramu, jeśli istnieje wartościowanie wyznaczające podgraf będący interpretacją tej formuły istniejący w hipergrafie reprezentującym rozważany diagram. Na przykład, relacja bezpośredniej dostępności pomiędzy dwoma pomieszczeniami $acc(x_1, x_2)$ poprzez ściany reprezentowane przez x_1 i x_2 jest spełniona, jeśli w hipergrafie istnieją dwa wierzchołki będące wartościowaniem x_1 i x_2 przypisane dwóm różnym hiperkrawędziom obiektowym i tej samej hiperkrawędzi relacyjnej etykietowanej acc. To oznacza, że: $(J, \omega) \models acc(x_1, x_2) \Leftrightarrow \exists\, v_1, v_2 \in V$ takie, że $\omega(x_1) = v_1$, $\omega(x_2) = v_2$, $v_1 \in t(e_1)$, $v_2 \in t(e_2)$, gdzie $e_1, e_2 \in E_C$ i $\exists\, e_3 \in E_R$ takie, że $v_1, v_2 \in t(e_3)$, $lb_R(e_3) = acc$, gdzie E_C, E_R, t i lb_R są dla danego hipergrafu odpowiednio zbiorami hiperkrawędzi obiektowych, relacyjnych, odwzorowaniem przypisującym hiperkrawędziom ciągi wierzchołków docelowych i funkcją etykietowania hiperkrawędzi relacyjnych.

Przykład 6.5: rozkłady pomieszczeń
Formuła $acc(x_1, x_2)$ zachodzi dla diagramu przedstawionego na rys. 6.1, na przykład przy wartościowaniu $\omega(x_1) = Be.2$, $\omega(x_2) = Ba.4$, które wyznacza podgraf hipergrafu przedstawiony na rys. 6.4a, a formuła $adj(x_1, x_2)$ zachodzi dla tego diagramu, na przykład przy wartościowaniu $\omega(x_1) = Ba.3$, $\omega(x_2) = Gr.1$, które wyznacza podgraf hipergrafu pokazany na rys. 6.4b. ◊

Atomowa formuła postaci ${}_P(t_1,\ldots, t_n)$ lub $t_1 = t_2$, gdzie t_1 i t_2 są termami rodzaju D, wyraża ograniczenia wiążące wartości atrybutów obiektów projektowych reprezentowanych przez komponenty diagramu. Wartościowanie termów powstałych przez zastosowanie funkcji odpowiadających atrybutom do termów rodzaju O przypisuje im odpowiednie wartości przechowywane w atrybutach wierzchołków i hiperkrawędzi obiektowych hipergrafu reprezentującego diagram projektowy. Na przykład, formuła $window\_number(x_1) + window\_number(x_2) = 3$ oznacza, że ściany reprezentowane przez x_1 i x_2 zawierają 3 okna i jest spełniona, jeśli istnieją dwa wierzchołki będące wartościowaniem x_1 i x_2, takie że suma wartości ich atrybutów *window_number* jest równa 3, a formuła $area(t_1) < area(t_2)$ jest speł-

niona, jeśli istnieją dwie hiperkrawędzie obiektowe będące wartościowaniem termów t_1 i t_2, takie że atrybut *area* pierwszej z nich ma mniejszą wartość niż ten sam atrybut drugiej hiperkrawędzi.

6.3. Logiczny model wnioskowania z wykorzystaniem hipergrafów

Rozwój rozwiązania projektowego, a także dynamika procesu projektowego wynikająca ze zmieniających się wymagań projektowych, nowych pomysłów i intencji projektanta, wymaga ciągłego wprowadzania zmian w strukturach projektów, które są odzwierciedlane w reprezentujących je diagramach projektowych. Każdej operacji wykonywanej na obiektach projektowych odpowiada zbiór akcji fizycznych, takich jak rysowanie czy usuwanie wielokątów i odcinków, wykonywanych na diagramie projektowym. Istota wnioskowania diagramowego polega na wyciąganiu wniosków dotyczących zmodyfikowanego diagramu (np. o dostępności określonych pomieszczeń) otrzymanego poprzez zastosowanie akcji projektowych. Akcje te modyfikując diagramy projektowe powodują jednocześnie automatyczne zmiany zarówno struktury i atrybutów hierarchicznych hipergrafów reprezentujących diagramy, jak i atomowych formuł opisujących diagramy. Struktura relacyjna przypisująca elementy hipergrafów formułom logicznym opisującym diagramy projektowe pozwala śledzić zmiany zachodzące w projektach na skutek akcji projektowych i wnioskować o własnościach diagramów.

Każdy nowy diagram powinien spełniać kryteria projektowe, które są istotne na tym etapie procesu projektowego. Dzięki strukturze relacyjnej wiedza zawarta w hierarchicznym hipergrafie reprezentującym diagram jest tłumaczona na atomowe formuły logiczne, opisujące ten diagram. Struktura ta pozwala także znaleźć wartościowania spełniające formuły atomowe, jak i zbudowane z nich formuły logiczne ze zbioru $\mathcal{F}$, określające własności diagramów projektowych będących elementami diagramowego języka projektowania. Wymagania i ograniczenia projektowe również mogą być wyrażone w postaci formuł logiki pierwszego rzędu zdefiniowanych nad tym samym słownikiem co formuły z $\mathcal{F}$ i z takim samym dopasowaniem ontologicznym tego słownika do pojęć konceptualizacji projektowej. Moduł wnioskujący będący częścią systemu wspomagającego projektanta dokonuje oceny rozwiązania reprezentowanego przez diagram projektowy poprzez sprawdzenie, czy formuły opisujące wygenerowany diagram są zgodne z formułami określającymi kryteria projektowe. Sprawdzenie spełnialności wymagania projektowego polega na znalezieniu wartościowania formuły ze zbioru $\mathcal{F}$, przy którym jest ona zgodna z formułą wyrażającą to wymaganie.

Niech Ψ będzie zbiorem formuł reprezentujących wymagania projektowe. Formuła $\varphi \in \Psi$ zachodzi dla danego diagramu, jeśli istnieje atomowa formuła równoważna φ, która

zachodzi dla tego diagramu lub można znaleźć formułę ϕ z $\mathcal{F}$, z której, przy odpowiednim wartościowaniu, można wywnioskować φ.

Niech J będzie $\mathcal{V}$- strukturą relacyjną zdefiniowaną dla projektowania diagramowego.

Definicja 6.5:
Niech $\varphi \in \Psi$ będzie formułą definiującą wymaganie projektowe, a $\mathcal{B}$ zbiorem atomowych formuł zachodzących dla diagramu projektowego Dg.

φ* jest spełnione przez *Dg $\Leftrightarrow \exists\, b \in \mathcal{B}$ takie, że $\varphi = b$ lub $\exists\, \phi \in \mathcal{F}$ i wartościowanie ω takie, że $(J, \omega) \models \phi \wedge \phi \Rightarrow \varphi$. □

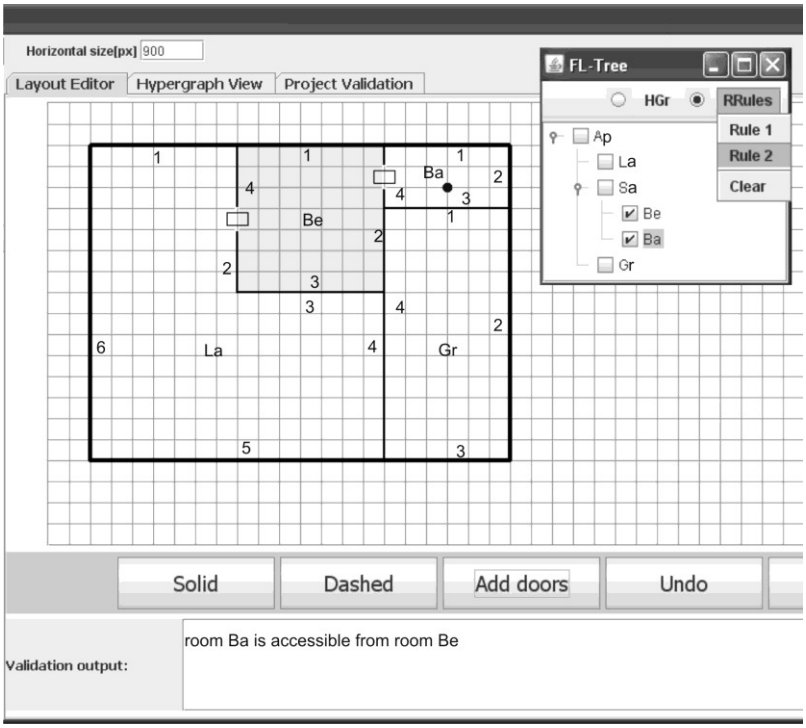
Dla danego hierarchicznego hipergrafu G i-ty element sekwencji wierzchołków zewnętrznych uchwytu indukowanego przez hiperkrawędź obiektową e będzie oznaczany przez $lb_O(blg^J(i)).i$, gdzie $lb_O(blg^J(i))$ jest etykietą hiperkrawędzi e, dla której i-ty element jest wierzchołkiem docelowym (tzn. $blg^J(i) = e$).

Przykład 6.6: rozkłady pomieszczeń
W przypadku projektowania rozkładów pomieszczeń, po każdym wykonaniu operacji *divide*, która dodaje do struktury projektu obiekty odpowiadające nowym komponentom diagramu, dla każdego nowego obszaru lub pomieszczenia sprawdzana jest jego dostępność z innego pomieszczenia lub obszaru. Dostępność obiektu Lb_1 z obiektu Lb_2 poprzez wspólną ścianę jest opisywana formułą $acc(x_1, x_2)$, gdzie $blg(x_1) = Lb_1$, $blg(x_2) = Lb_2$. Znalezienie odpowiedniego wartościowania dla formuły postaci $acc(x_1, x_2)$ wyznacza podgraf wygenerowanego hipergrafu złożony z uchwytów o etykietach Lb_1 i Lb_2, których wierzchołki odpowiadające tej samej ścianie są połączone hiperkrawędzią relacyjną o etykiecie *acc*. Projektant jest informowany o wszystkich pomieszczeniach, dla których ta formuła nie jest spełniona, czyli takich, które są dostępne tylko z zewnątrz. W hipergrafie z rys. 3.2 odpowiedni podgraf nie zostanie znaleziony jedynie dla hiperkrawędzi obiektowej etykietowanej *Gr*, która reprezentuje garaż. Oznacza to, że garaż nie ma połączenia z żadnym z pomieszczeń mieszkania i jest dostępny tylko z zewnątrz domu.

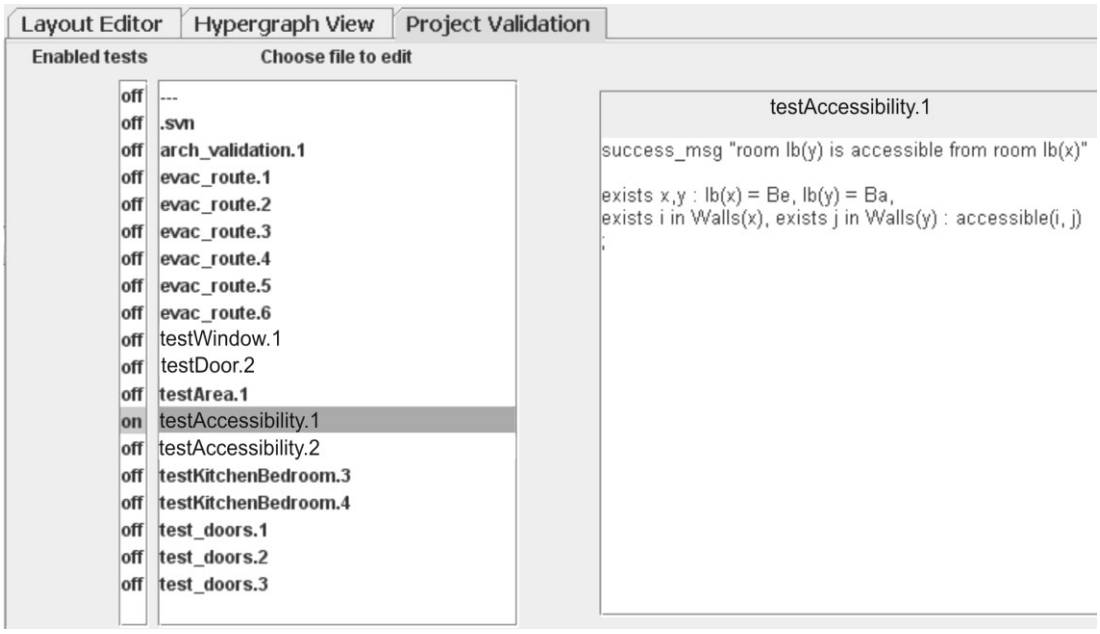
Istnienie odpowiedniego wartościowania powyższej formuły jest istotne także w przypadku, gdy nie jest wskazane, aby w rozkładzie pomieszczeń określone dwa pomieszczenia były bezpośrednio dostępne. Wartościowanie postaci $\omega(x_1) = Wc.i$ i $\omega(x_2) = Kt.j$ (gdzie wierzchołek będący wartością $\omega(x)$ dla zmiennej x został zastąpiony swoją etykietą) oznacza, że toaleta jest bezpośrednio dostępna z kuchni. Istnienie takiej relacji w projekcie wymaga akceptacji użytkownika.

Przykład sprawdzania bezpośredniej dostępności pomiędzy sypialnią (*Be*) i łazienką (*Ba*) jest przedstawiony na rys. 6.5. Wybór sprawdzanych pomieszczeń jest możliwy w oknie hierarchii pokazanym po prawej stronie, a wynik testu dostępności przedstawionego na rys. 6.6

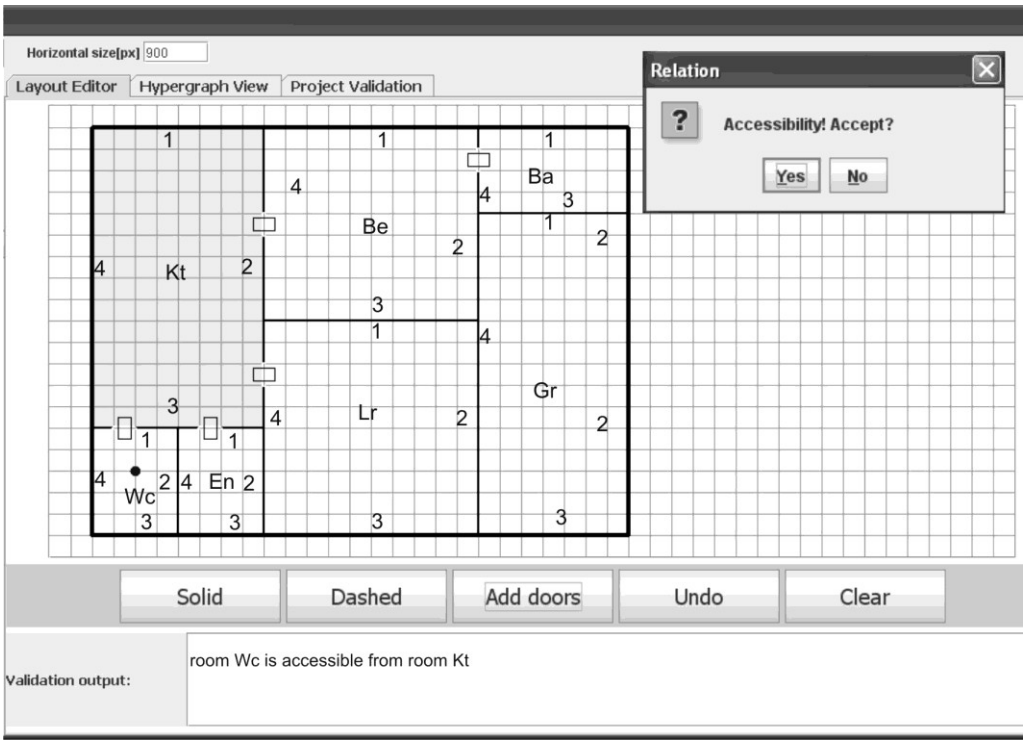
jest pokazany w dolnej części okna (rys. 6.5). Ponadto, łazienka jako pomieszczenie dostępne z sypialni oznaczone jest kropką, a sypialnia innym kolorem niż pozostałe pomieszczenia. Prośba o akceptację dostępności pomiędzy kuchnią i toaletą przez użytkownika (w kroku projektowym przedstawionym na rys. 4.6a) jest pokazana na rys. 6.7.



Rys. 6.5. Sprawdzanie bezpośredniej dostępności pomiędzy dwoma wybranymi pomieszczeniami w programie *HSSDR*
Fig. 6.5. Checking the accessibility between two chosen rooms in *HSSDR* program



Rys. 6.6. Test sprawdzający dostępność między wybranymi pomieszczeniami w programie *HSSDR*
Fig. 6.6. A test checking the accessibility between chosen rooms in *HSSDR* program



Rys. 6.7. Sprawdzanie poprawności relacji dostępności pomiędzy kuchnią i toaletą w programie *HSSDR*

Fig. 6.7. Validating the accessibility relation between the kitchen and toilet in *HSSDR* program

W podobny sposób można sprawdzić, czy wszystkie pomieszczenia w mieszkaniu są bezpośrednio dostępne z holu. Wymaganie to ma postać formuły: $\forall\ t_1 \in room\colon t_1 \neq Hl \ \exists\ t_2 = Hl \wedge \exists\ x_i \in wall\colon blg(x_i) = t_1 \wedge \exists\ x_k \in wall\colon blg(x_k) = t_2 \wedge acc(x_k, x_i)$.

Sprawdzenie spełnialności tej formuły przez dany diagram polega na znalezieniu odpowiedniego wartościowania dla koniunkcji formuł wyrażających bezpośrednią dostępność między holem i innymi pomieszczeniami. Komunikat przekazywany użytkownikowi przez program *HSSDR* [GBPG09] to "*all rooms are accessible from room Hl*" lub jego zaprzeczenie. ◊

Przykład 6.7: rozkłady pomieszczeń

Niech $\Psi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ zawiera cztery formuły opisujące wymagania projektowe. Formuła $\varphi_1 = acc(x_1, x_2)$, gdzie $blg(x_1) = Dr$, $blg(x_2) = Kt$, mówi, że jadalnia powinna być bezpośrednio dostępna z kuchni, $\varphi_2 = \neg acc(x_1, x_2)$, gdzie $blg(x_1) = Kt$, $blg(x_2) = Wc$, mówi, że toaleta nie powinna być bezpośrednio dostępna z kuchni, $\varphi_3 = acc(x_1, x_2)$, gdzie $blg(x_1) = Hl$, $blg(x_2) = Gr$, mówi, że garaż powinien być dostępny z holu, a φ_4 jest postaci $acc(x_1, x_2, x_3, x_4)$, $blg(x_1) = Lr$, $blg(x_4) = Ba$, $blg(x_2) = blg(x_3)$ i oznacza, że łazienka jest dostępna z salonu poprzez inne pomieszczenie.

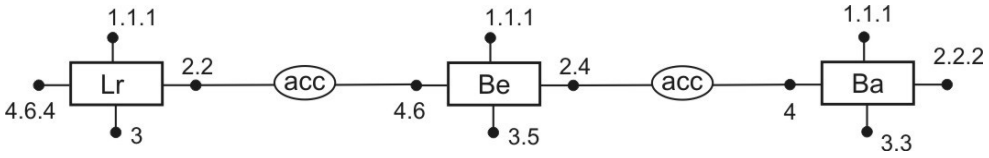
Sprawdźmy, czy formuły ze zbioru Ψ są spełnione przez diagram projektowy przedstawiony na rys. 6.3. W diagramie tym nie ma komponentu reprezentującego obiekt o nazwie *Dr*, a więc φ_1 nie jest spełnione. Nie istnieją też wartościowania spełniające formułę $acc(x_1, x_2)$, gdzie x_1 i x_2 byłyby odpowiednio postaci *Kt.i* i *Wc.j*, oraz *Hl.i* i *Gr.j*, więc formuła φ_2 jest spełniona, a formuła φ_3 nie jest. Natomiast dla formuły $acc(x_1, x_2) \wedge acc(x_3, x_4) \wedge blg(x_2) = blg(x_3) \Rightarrow acc(x_1, x_2, x_3, x_4)$ można znaleźć wartościowanie zmiennych postaci $\omega(x_1) = Lr.2$, $\omega(x_2) = Be.4$, $\omega(x_3) = Be.2$, $\omega(x_4) = Ba.4$, przy którym φ_4 jest spełnione.

Dla tej samej formuły i diagramu przedstawionego na rys. 6.7 można znaleźć trzy różne wartościowania zmiennych (pierwsze postaci: $\omega(x_1) = Lr.4$, $\omega(x_2) = Kt.2$, $\omega(x_3) = Kt.2$, $\omega(x_4) = Be.4$, drugie postaci: $\omega(x_1) = Lr.4$, $\omega(x_2) = Kt.2$, $\omega(x_3) = Kt.3$, $\omega(x_4) = En.1$, trzecie postaci: $\omega(x_1) = Lr.4$, $\omega(x_2) = Kt.2$, $\omega(x_3) = Kt.3$, $\omega(x_4) = Wc.1$), z których żadne nie spełnia formuły φ_4. ◊

Ograniczeniom projektowym wyrażanym jako formuły logiczne odpowiadają określone warunki, jakie powinny spełniać hipergrafy, jak np. istnienie w hipergrafie pewnej ścieżki. Sprawdzenie warunku dotyczącego dostępności pewnego pomieszczenia o etykiecie Lb_n z pomieszczenia o etykiecie Lb_1 poprzez inne pomieszczenia wymaga znalezienia wartościowania ω formuły $acc(x_1, x_2) \wedge \ldots \wedge acc(x_{n-1}, x_n) \Rightarrow acc(x_1, \ldots, x_n)$, gdzie $blg(x_i) = blg(x_{i+1})$, $i = 2, 4, \ldots, n-2$, $blg(x_1) = Lb_1$, $blg(x_n) = Lb_n$, gdyż zmienna x_1 reprezentuje jedną ze ścian pomieszczenia Lb_1, a zmienna x_n reprezentuje ścianę pomieszczenia Lb_n. Wartościowanie to powinno spełniać warunki $blg^J(\omega(x_1)) = e^o_1$, gdzie e^o_1 jest hiperkrawędzią obiektową o etykiecie Lb_1 i $blg^J(\omega(x_n)) = e^o_n$, gdzie e^o_n jest hiperkrawędzią obiektową o etykiecie Lb_n. Wartościowanie takie odpowiada istnieniu w hipergrafie ścieżki od wierzchołka przypisanego hiperkrawędzi obiektowej o etykiecie Lb_1 do wierzchołka przypisanego hiperkrawędzi obiektowej o etykiecie Lb_n. Ścieżka ta ma postać $e^o_1 e^r_1 e^o_2 \ldots e^r_{n-1} e^o_n$, gdzie e^o_i, $i = 1,\ldots, n$, są różnymi hiperkrawędziami obiektowymi, e^r_j, $j = 1,\ldots, n-1$, są hiperkrawędziami relacyjnymi etykietowanymi *acc*, e^o_1 jest hiperkrawędzią etykietowaną Lb_1, e^o_n jest hiperkrawędzią etykietowaną Lb_n, oraz $\forall e^o_i e^r_i e^o_{i+1} \; \exists v_j, v_k$ takie, że $v_j \in t(e^o_i) \cap t(e^r_i)$, $v_k \in t(e^r_i) \cap t(e^o_{i+1})$, gdzie t jest odwzorowaniem wyznaczającym sekwencje wierzchołków docelowych dla hiperkrawędzi. Hiperkrawędzie relacyjne należące do ścieżki wraz z uchwytami indukowanymi przez hiperkrawędzie obiektowe występujące w tej ścieżce i określone przez funkcję blg^J zastosowaną do wierzchołków wyznaczonych przez znalezione wartościowanie zmiennych tworzą podgraf hipergrafu reprezentującego diagram.

Przykład 6.8: rozkłady pomieszczeń

Dostępność łazienki z salonu odpowiada istnieniu w hipergrafie ścieżki od wierzchołka reprezentującego jedną ze ścian salonu do wierzchołka reprezentującego jedną ze ścian łazienki. Ścieżka ta ma postać *Lr acc Be acc Ba*, gdzie kolejne hiperkrawędzie zostały zastąpione ich etykietami. Hiperkrawędzie relacyjne wraz z uchwytami indukowanymi przez hiperkrawędzie obiektowe występujące w tej ścieżce tworzą podgraf hipergrafu z rys. 3.2 przedstawiony na rys. 6.8. ◊



Rys. 6.8. Podgraf reprezentujący ścieżkę pomiędzy salonem a łazienką
Fig. 6.8. A subgraph representing a path between the living-room and bathroom

Wykorzystanie w formułach logicznych funkcji arytmetycznych i predykatów relacyjnych operujących na wartościach przypisanych atrybutom termów reprezentujących obiekty projektowe pozwala wnioskować o semantycznych własnościach diagramów.

Przykład 6.9: rozkłady pomieszczeń

W przypadku projektowania rozkładów pomieszczeń każdej hiperkrawędzi obiektowej są przypisane atrybuty *area* i *position*, których wartości określają powierzchnię i lokalizację obszaru lub pomieszczenia reprezentowanego przez tę hiperkrawędź. Wierzchołkom hipergrafu są przypisane atrybuty *order*, *orientation*, *length*, *position*, *window_number* i *door_number*. Określają one kolejność, orientację, długość i lokalizację ścian oraz liczbę okien i drzwi, jakie znajdują się w ścianach.

W omawianym systemie wspomagania projektowania i wnioskowania *HSSDR* istnieje między innymi możliwość sprawdzenia, czy:

- powierzchnia wszystkich pokoi jest co najmniej równa zadanej stałej z (*testArea*1: $\forall\ t \in room\ area(t) \geq z$). Jeśli z jest równe 8, to dla diagramu projektowego przedstawionego na rys. 6.3 można znaleźć wartościowanie $\omega(t) = Wc$ spełniające formułę $\exists\ t: area(t) = 3.5$ i wartościowanie $\omega(t) = En$ spełniające formułę $\exists\ t: area(t) = 7.5$, wobec czego powyższe wymaganie nie jest spełnione,
- w danym pomieszczeniu istnieje co najmniej jedno okno. Dla kuchni z diagramu pokazanego na rys. 6.3 formuła odpowiadająca temu warunkowi – *testWindow*1: $\exists\ x \in wall: blg(x) = Kt \wedge window\_number(x) \geq 1$ nie jest spełniona przy żadnym wartościowaniu zmiennej x,

- dane pomieszczenie posiada dwoje drzwi. Dla kuchni z rys. 6.3 formuła odpowiadająca temu warunkowi – *testDoor*2: $\exists\, x \in wall\_pass$: $blg(x) = Kt \wedge door\_number(x) = 2$ nie jest spełniona, natomiast dla diagramu z rys. 6.7 można znaleźć wartościowania $\omega(x) = Kt.2$ lub $\omega(x) = Kt.3$ spełniające tę formułę,
- kuchnia ma co najmniej dwoje drzwi na dwóch różnych ścianach (*testDoor*3: $\exists\, x_1, x_2 \in wall\_pass$: $x_1 \neq x_2$, $blg(x_1) = blg(x_2) = Kt \wedge door\_number(x_1) + door\_number(x_2) \geq 2$). Dla diagramu z rys. 6.7 można znaleźć wartościowanie postaci $\omega(x_1) = Kt.2$ spełniające formułę $\exists\, x_1 \in wall\_pass$: $blg(x_1) = Kt \wedge door\_number(x_1) = 2$ oraz wartościowanie postaci $\omega(x_2) = Kt.3$ spełniające formułę $\exists\, x_2 \in wall\_pass$: $blg(x_2) = Kt \wedge door\_number(x_2) = 2$. Z koniunkcji tych dwóch formuł wynika, że powyższe wymaganie jest spełnione. Jest ono również spełnione dla diagramu z rys. 6.3 przy wartościowaniu $\omega(x_1) = Kt.1$ i $\omega(x_2) = Kt.2$.

System wspomagający projektowanie sprawdza też, czy można umieścić wszystkie zaprojektowane okna na danej ścianie poprzez porównanie wartości atrybutu *window_number* przypisanego wierzchołkowi reprezentującemu tę ścianę i pomnożonego przez szerokość okien z wartością atrybutu *length* dla tej ściany. System sprawdza także, czy okna dostarczą do każdego pokoju występującego w diagramie odpowiedniej ilości światła. W tym celu są sumowane wartości atrybutów *window_number* przypisanych wszystkim wierzchołkom połączonym z hiperkrawędzią reprezentującą dany pokój pomnożone przez powierzchnie okien, a otrzymana liczba jest porównywana z określonymi normami architektonicznymi. System podpowiada projektantowi, że potrzebny jest podział pomieszczenia lub dodanie elementów podtrzymujących strop, w przypadku gdy wartość atrybutu *area* jest zbyt duża względem rozważanego pomieszczenia. System sugeruje także projektantowi (na podstawie wartości atrybutu *door_number*), że hol ze zbyt dużą liczbą drzwi powinien być podzielony na dwa mniejsze pomieszczenia. ◊

Przykład 6.10: krzesła

W przypadku projektowania krzeseł hiperkrawędziom obiektowym są przypisane atrybuty *width*, *height*, *depth* określające rozmiary reprezentowanych przez nie komponentów. Dzięki temu odpowiednie warunki zdefiniowane na podstawie wartości atrybutów pozwalają obliczyć środek ciężkości krzesła, sprawdzić, czy oparcie oraz nogi krzesła nie są zbyt niskie lub zbyt wysokie, a także sprawdzić, czy siedzisko krzesła nie jest zbyt wąskie. ◊

Przykład 6.11: wieże przesyłowe

Przykładowe zdanie opisujące diagramy projektowe wież przesyłowych, $\forall\, t \in truss$ $width(t) < 5$ mówi, że szerokość żadnego ze skratowań nie wynosi więcej niż 5 metrów. System wspomagający projektowanie wież kratowych sprawdza, czy wygenerowana wieża nie

jest zbyt szeroka. W tym celu są sumowane wartości atrybutów *width* przypisanych wszystkim hiperkrawędziom obiektowym reprezentującym skratowania poziome, a następnie otrzymana liczba jest porównywana z wymaganą szerokością. Dla diagramu przedstawionego na rys. 2.3b wartościowanie termów $t_1,..., t_7$ postaci $\omega(t_i) = T_i$, $i = 1,\ldots, 7$, spełnia formułę $width(t_1) + \ldots + width(t_7) < 15$, ponieważ $width(T_1) = width(T_4) = width(T_7) = 3$, $width(T_2) = width(T_6) = 1$, $width(T_3) = width(T_5) = 1.5$. W analogiczny sposób system sprawdza wysokość wieży. Formuła $height(t_1) + height(t_2) + height(t_3) + height(t_4) < 23$ jest spełniona przy wartościowaniu $\omega(t_1) = Te_1$, $\omega(t_2) = Te_2$, $\omega(t_3) = Te_4$, $\omega(t_4) = Te_6$, ponieważ $height(Te_1) = height(Te_2) = 5$, $height(Te_4) = height(Te_6) = 6$. System może także obliczyć rozkład obciążeń w zaprojektowanej strukturze szkieletowej na podstawie atrybutów opisujących przekroje prętów i materiał, z jakiego mają być wykonane. ◊

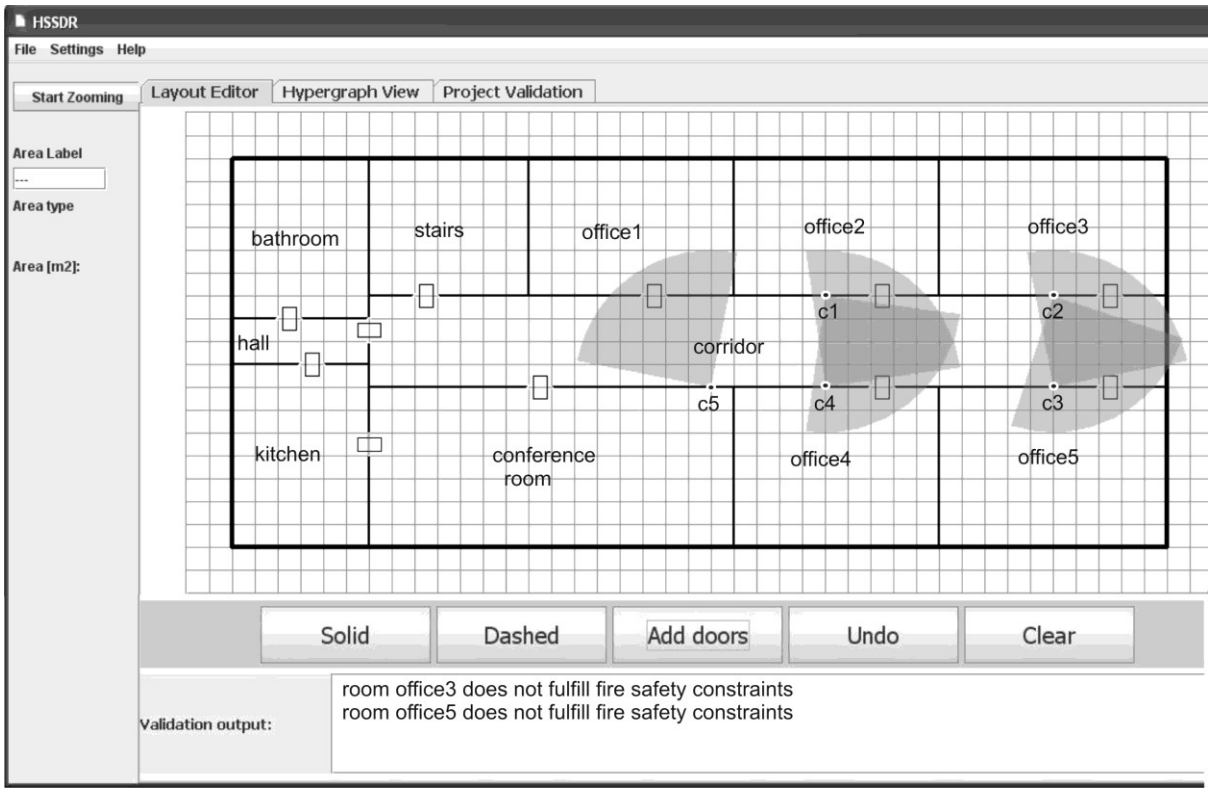
System wspomagający projektowanie rozkładów pomieszczeń można wyposażyć w warunki sprawdzające poprawność rozmieszczenia czujników ruchu i kamer monitorujących poszczególne piętra budynku [BGGP11, GrŚl11a]. Bezpieczeństwo budynku będzie zapewnione, jeśli określone drogi pomiędzy pomieszczeniami bądź drzwi prowadzące do pomieszczeń będą się znajdowały w polu widzenia czujników lub kamer.

Przykład 6.12: rozkłady pomieszczeń

W przypadku rozważania monitoringu projektowanego rozkładu pomieszczeń wprowadzamy dodatkowe obiekty projektowe reprezentujące kamery oraz relację *on* reprezentującą umieszczenie kamery na ścianie. Rozkład pomieszczeń z pięcioma kamerami umieszczonymi na ścianach korytarza jest przedstawiony na rys. 6.9. Czarne kropki etykietowane $c_1,.., c_5$ reprezentują kamery, a szare wycinki kół odpowiadają polu widzenia kamer i są automatycznie generowane przez system *HSSDR* dla wstawianych kamer na podstawie wartości ich atrybutu *range* określanego przez użytkownika.

Załóżmy, że warunkiem zapewnienia bezpieczeństwa jest monitorowane przez kamery wejść do wszystkich pomieszczeń biurowych. Wymaganie to można wyrazić w języku logiki pierwszego rzędu za pomocą formuły postaci: $\forall\, t = office\ \exists\, t' \in camera$: $observed(t, t')$, gdzie predykat *observed* zachodzi, jeśli pomieszczenie t jest pod obserwacją kamery t'. W celu sprawdzenia, czy powyższa formuła jest spełniona przez diagram z rys. 6.9, predykat $observed(t, t')$ jest rozwijany do postaci $\exists\, x, x' \in wall$, $blg(x) = t \wedge blg(x') = corridor \wedge on(x', t') \wedge inrange(range(t'), loc\_door(x))$. Predykat $inrange(range(t'), loc\_door(x))$, gdzie atrybut *loc_door* określa lokalizację drzwi na ścianie, jest spełniony, jeśli drzwi ściany x są w zasięgu kamery t', co oznacza, że prostokąt reprezentujący drzwi musi znajdować się

wewnątrz wycinka koła reprezentującego zasięg tej kamery. Na rys. 6.9 drzwi do każdego z pomieszczeń biurowych (*office*1-*office*5) są w zasięgu jednej z kamer. ◊



Rys. 6.9. Rozkład pomieszczeń z pięcioma kamerami otrzymany w programie *HSSDR*
Fig. 6.9. A floor layout with five cameras obtained in *HSSDR* program

Odpowiednie sformułowanie warunków dotyczących diagramów pozwala sprawdzić wymagania przeciwpożarowe dla projektowanych budynków [GBPG09] i wyznaczyć optymalne ścieżki ewakuacyjne. Polskie przepisy przeciwpożarowe mówią, że każde pomieszczenie powinno mieć dostęp do drogi ewakuacyjnej przez nie więcej niż trzy inne pomieszczenia. Ponadto, żadna droga ewakuacyjna prowadząca do schodów lub drzwi zewnętrznych nie może być dłuższa niż 30 metrów.

Przykład 6.13: rozkłady pomieszczeń
Wymaganie przeciwpożarowe, mówiące, że wszystkie pomieszczenia powinny mieć dostęp do drogi ewakuacyjnej przez nie więcej niż trzy inne pomieszczenia, jest spełniony dla rozkładu pomieszczeń z rys. 6.9, jeśli dla każdego z pomieszczeń zachodzi jedna z poniższych formuł:

1. $\forall\ t \in room,\quad t \neq corridor \wedge t \neq staircase \quad \exists\ t' \in room,\quad t' = corridor \vee t' = staircase$: $acc(t, t')$, gdzie $acc(t, t') \Leftrightarrow \exists\ x, x' \in wall, blg(x) = t \wedge blg(x') = t'$: $acc(x, x')$,
2. $\forall\ t \in room,\quad t \neq corridor \wedge t \neq staircase \quad \exists\ t_1 \in room,\quad t_1 \neq corridor \wedge t_1 \neq staircase \wedge \exists\ t' \in room, t' = corridor \vee t' = staircase$: $acc(t, t_1) \wedge acc(t_1, t')$,
3. $\forall\ t \in room,\quad t \neq corridor \wedge t \neq staircase \quad \exists\ t_1, t_2 \in room,\quad t_1, t_2 \neq corridor \wedge t_1, t_2 \neq staircase \wedge \exists\ t' \in room,\ t' = corridor \vee t' = staircase$: $acc(t, t_1) \wedge acc(t_1, t_2) \wedge acc(t_2, t')$,
4. $\forall\ t \in room,\quad t \neq corridor \wedge t \neq staircase \quad \exists\ t_1, t_2, t_3 \in room,\quad t_1, t_2, t_3 \neq corridor \wedge t_1, t_2, t_3 \neq staircase \quad \wedge \exists\ t' \in room,\quad t' = corridor \vee t' = staircase$: $acc(t, t_1) \wedge acc(t_1, t_2) \wedge acc(t_2, t_3) \wedge acc(t_3, t')$.

Dla pomieszczeń z rys. 6.9 o etykietach *bathroom* i *kitchen* jest spełniony warunek 2, natomiast dla pozostałych pomieszczeń zachodzi warunek 1. Sprawdzenie normy, określającej, że wszystkie drogi ewakuacyjne prowadzące do schodów nie są dłuższe niż 30 metrów [GrŚ11a], wymaga dodania do słownika formuł logicznych symbolu $dist(\Re^2 \times \Re^2)$: $\Re$ funkcji obliczającej odległość między dwoma punktami. Warunek ten można wówczas zapisać za pomocą formuły: $\forall\ x \in wall,\ blg(x) = corridor,\ door\_number(x) \geq 1 \quad \exists\ x' \in wall,\ blg(x') = staircase \wedge door\_number(x') \geq 1$: $\forall\ i \in \{1,\ldots, door\_number(x)\},\quad \exists\ j \in \{1, \ldots, door\_number(x')\}$: $dist(loc\_door_i(x), loc\_door_j(x')) \leq 30$, gdzie atrybut ściany *loc_door* określa współrzędne drzwi umieszczonych na tej ścianie. Formuła ta nie jest spełniona przez drzwi prowadzące do pomieszczeń biurowych o etykietach *office*3 i *office*5, a komunikat o tym jest wyświetlony na dole ekranu pokazanego na rys. 6.9. Sytuacje tę można skorygować przesuwając drzwi w kierunku schodów lub schody w kierunku środka budynku.

W podobny sposób można sprawdzić, czy odległość łazienki (*Ba*) przedstawionej na diagramie z rys. 6.3 od wejścia znajdującego się na ścianie *En*.3 mieści się w podanym zakresie. Należy najpierw znaleźć wartościowanie spełniające formułę wyrażającą dostępność między łazienką a przedsionkiem $acc(x_1,\ldots, x_n)$, gdzie $blg(x_i) = blg(x_{i+1}),\ i = 2, 4,\ldots, n-2$, $blg(x_1) = Ba,\ blg(x_n) = En$, a następnie obliczyć $dist(loc\_door(Ba.4), loc\_door(Be.4)) + dist(loc\_door(Ba.4), loc\_door_2(Lr.2)) + dist(loc\_door_2(Lr.2), loc\_door(En.1)) + length(En.4)$. ◊

Informacje o rozmieszczeniu pomieszczeń na poszczególnych piętrach budynku zakodowane w atrybutowanej hierarchicznej hipergrafowej reprezentacji pozwalają także wywnioskować fakty użyteczne do nawigacji mobilnych robotów umieszczonych w budynku

[CLHK05, ThBF05]. Robot poruszający się po budynku dzięki hipergrafowej reprezentacji zna nie tylko rozmieszczenie poszczególnych pomieszczeń, ale także ich funkcje i może wyznaczyć ścieżki prowadzące do określonych pokoi. Odpowiednie wartościowanie, które spełnia formuły logiczne wyrażające wymagania projektowe, pozwala systemowi znaleźć zarówno najkrótsze drogi łączące pokoje, jak i drogi alternatywne oraz sprawdzić dostępność istniejących dróg [GrŚl11]. Znając swoje rozmiary i promień skrętu, robot testując wartości atrybutów określających szerokość pomieszczeń może sprawdzić przed rozpoczęciem marszruty, czy będzie w stanie przejechać wyznaczoną drogą.

Przykład 6.14: rozkłady pomieszczeń

Załóżmy, że robot sprzątający ma po wyjściu z pomieszczenia biurowego *office*4 z rys. 6.9 posprzątać salę konferencyjną. Istnienie odpowiedniej drogi między tymi pomieszczeniami można wyrazić formułą $\exists\, x_1,\ldots, x_n \in wall \;\; blg(x_1) = office4, \;\; blg(x_n) = conference\_room$, $acc(x_1, x_2) \wedge ... \wedge acc(x_{n-1}, x_n)$. Moduł wnioskujący systemu wspomagającego projektowanie znajdzie dwa wartościowania zmiennych spełniające tę formułę:

1. $\omega(x_1) = office4.1$, $\omega(x_2) = \omega(x_3) = corridor.3$, $\omega(x_4) = conference\_room.1$,
2. $\omega(x_1) = office4.1$, $\omega(x_2) = corridor.3$, $\omega(x_3) = corridor.4$, $\omega(x_4) = hall.2$, $\omega(x_5) = hall.3$, $\omega(x_6) = kitchen.1$, $\omega(x_7) = kitchen.2$, $\omega(x_8) = conference\_room.4$.

Pierwsze z tych wartościowań wyznacza drogę prowadzącą przez korytarz, natomiast drugie wyznacza drogę prowadzącą przez pomieszczenia etykietowane *corridor*, *hall* i *kitchen*. ◊

6.4. Podsumowanie

W tym rozdziale został omówiony logiczny model wnioskowania o poprawności projektów z wykorzystaniem wewnętrznej reprezentacji diagramów projektowych w postaci hierarchicznych hipergrafów rozmieszczenia i formuł logicznych pierwszego rzędu opisujących własności tych diagramów. Został zdefiniowany wielorodzajowy słownik formuł logicznych oraz dopasowanie ontologiczne symboli z tego słownika do pojęć konceptualizacji projektowej, zapewniające taką interpretację syntaktyki formuł, która umożliwia reprezentację wiedzy dotyczącej obiektów projektowych odpowiadających komponentom diagramów. Następnie nad zdefiniowanym słownikiem został stworzony zbiór formuł charakteryzujących diagramy projektowe. Została przedstawiona struktura relacyjna, która przypisując atomy hipergrafów i wartości ich atrybutów elementom składowym formuł logicznych wyznacza semantykę tych formuł. Wprowadzone wartościowanie zmiennych i termów na strukturze relacyjnej dopełnia interpretację formuł języka logicznego. Interpretacja ta umożliwia pozyskiwanie wiedzy projektowej dotyczącej diagramów i śledzenie zmian zachodzących w strukturach projektów w wyniku wykonywania kolejnych akcji projektowych.

Rozważane formuły logiczne pozwalają wnioskować o projektach wizualizowanych jako diagramy. Zbiór formuł opisujących diagramy projektowe jest porównywany ze zbiorem formuł logicznych pierwszego rzędu wyrażających ograniczenia i wymagania projektowe. Sprawdzanie spełnialności formuł reprezentujących wymagania określone normami i przepisami projektowymi umożliwia systemowi projektowemu efektywne wspomaganie koncepcyjnej fazy projektowania. Logiczny mechanizm wnioskowania o jakości rozwiązań projektowych został zilustrowany na przykładach dotyczących projektowanych rozkładów pomieszczeń, krzeseł i wież przesyłowych.

7. AGENCI W PROJEKTOWANIU

W tym rozdziale zostanie przedstawione wykorzystanie agentów jako inteligentnych asystentów projektanta. Inteligentni agenci potrafią generować i modyfikować reprezentację środowiska na podstawie doświadczeń płynących z interakcji z projektantem i innymi agentami, jak również oceniać rozwiązania częściowe wygenerowane w trakcie procesu projektowego i sugerować projektantowi oryginalne rozwiązania [Wood99]. Agenci projektowi wykorzystujący hierarchiczne hipergrafowe gramatyki rozmieszczenia i potrafiący adaptować się do kryteriów projektowych zmieniających się w trakcie koncepcyjnego etapu projektowania zostali zdefiniowani przez Autorkę w pracy [Ślus08]. Prezentowany tu agentowy system projektowy został rozszerzony o logiczny mechanizm oceny zgodności rozwiązań z kryteriami projektowymi oparty na modalnym rachunku predykatów.

Przedstawiany w pracy system agentowy może być traktowany jako modularny współbieżny system rozwiązujący skomplikowane zadania projektowe. Architektura tego systemu opiera się na zbiorze inteligentnych agentów projektowych, którzy komunikując się ze sobą autonomicznie sterują rozwojem rozwiązań podzadań projektowych. Dynamicznie zmieniający się kontekst projektowy [Knig04] jest odzwierciedlany przez środowisko, w którym działają agenci, jak również predykaty opisujące kryteria projektowe.

Działanie każdego agenta projektowego jest modelowane z wykorzystaniem zbioru reguł hierarchicznej hipergrafowej gramatyki rozmieszczenia. Ponadto, każdy agent posiada bazę danych zawierającą informacje dotyczące podzadania, za które jest odpowiedzialny. Agenci tworzą swoje reprezentacje projektu w postaci atrybutowanych hierarchicznych hipergrafów. Dzięki takiej reprezentacji agenci mogą adaptować się do zmian zachodzących w specyfikacji problemu i realizować adaptacyjne poszukiwanie rozwiązań projektowych. Wspomaganie decyzji przez inteligentnych agentów w trakcie procesu projektowania wymaga od nich umiejętności wnioskowania z hipergrafów. Porównując predykaty wyrażające wymagania i ograniczenia projektowe z wiedzą projektową zakodowaną w hipergrafach agenci dokonują oceny rozwiązań częściowych reprezentujących aktualny stan projektowanego obiektu i na tej podstawie podejmują decyzje projektowe. Swoje przyszłe zachowanie planują poprzez wybór reguł gramatyki, które mają zostać zastosowane.

Zostanie zaprezentowana architektura systemu wieloagentowego wraz z przepływem informacji w takim systemie. Elementy języka hipergrafowego generowanego przez współpracujących agentów są interpretowane w postaci diagramów projektowych stanowiących podzbiór języka wizualnego wykorzystywanego przez użytkownika do komunikacji z systemem wspomagania projektowania. Zostaną przedstawione pojęcia poprawnego rozwiązania projektowego i rozwiązania zgodnego z określonymi kryteriami projektowymi. Do oceny rozwiązań zostanie wykorzystany wprowadzony w pracy mechanizm wnioskowania bazujący na modalnym rachunku predykatów. Przedstawione tu podejście zostanie zilustrowane na przykładzie projektowania domu jednorodzinnego, wspomaganego przez kilku współpracujących ze sobą agentów wykorzystujących hierarchiczne hipergrafowe gramatyki rozmieszczenia.

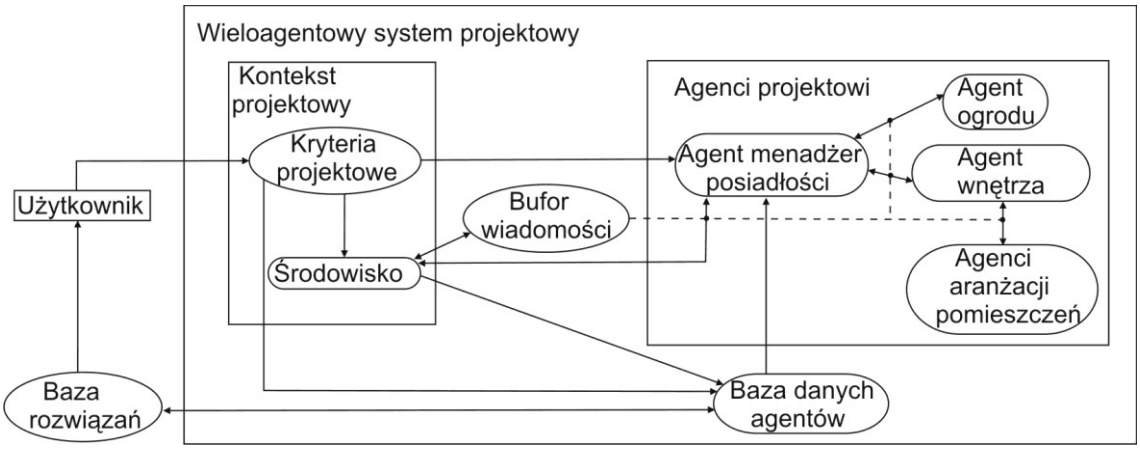
7.1. System agentów projektowych

Zaproponowany tu system wieloagentowy składa się z kilku rodzajów agentów równocześnie wykonujących różne zadania, kontekstu zawierającego środowisko agentów i określone kryteria projektowe oraz bufora przechowującego wiadomości.

Zastosowanie w procesie projektowym inteligentnych agentów będących asystentami projektanta jest zilustrowane na przykładzie projektowania funkcjonalnej, umeblowanej przestrzeni parterowego domu jednorodzinnego wraz z odpowiednio zaaranżowanym ogrodem.

Wyróżniony agent-menadżer jest odpowiedzialny za zaprojektowanie układu całej posiadłości, który byłby zgodny z określoną specyfikacją. Agent ten wywołuje dwóch innych agentów: jeden z nich projektuje wnętrze domu, natomiast drugi – ogród. Agent odpowiedzialny za zaprojektowanie umeblowanego rozkładu pomieszczeń domu posiada agentów-asystentów generujących rozkłady mebli i sprzętów dla różnego rodzaju pomieszczeń. W końcowym etapie projektowania agent-menadżer łączy projekty generowane przez samego siebie, agenta projektującego ogród i agenta projektującego dom w jedno rozwiązanie, które jest przesyłane do środowiska. Środowisko, które także jest traktowane jak agent, ocenia otrzymane rozwiązania i przechowuje je w globalnej bazie danych. Oprócz najlepszych rozwiązań przechowuje także kilka rozwiązań gorszych, które jednak na dalszym etapie mogą również prowadzić do korzystnych projektów. Rozwiązania z bazy danych mogą być wizualizowane w postaci modeli graficznych, zgodnie z określoną realizacją (def. 3.4).

Przepływ informacji w omawianym systemie jest pokazany na rys. 7.1. Przerywana linia przecinająca strzałki łączące agentów oznacza, że agenci i środowisko komunikują się ze sobą za pośrednictwem bufora.



Rys. 7.1. Przepływ informacji w omawianym systemie agentowym
Fig. 7.1. Information flow in the described agent system

System wieloagentowy efektywnie wspomaga projektowanie, gdyż każdy agent niezależnie i w sposób adaptacyjny poszukuje rozwiązań podzadania, za które jest odpowiedzialny. Wykorzystani tu agenci projektowi są wyposażeni w programowane hierarchiczne hipergrafowe gramatyki rozmieszczenia generujące reprezentacje rozwiązań w postaci atrybutowanych hierarchicznych hipergrafów i w bazy danych dotyczących podzadań, które razem reprezentują lokalną wiedzę projektową agentów. Przepływ informacji związany z wewnętrznymi procesami agenta (percepcja, planowanie i wykonywanie akcji projektowych, ocena aktualnego stanu projektu), który komunikuje się z innymi agentami poprzez bufor, jest pokazany na rys. 7.2. Nazwy w nawiasach oznaczają procesy i stany agenta, które będą wyspecyfikowane w definicji 7.1.

Na początku agent sprawdza bufor i odczytuje otrzymaną wiadomość. Następnie w procesie percepcji rozpoznaje typ wiadomości, która ma postać atrybutowanego hierarchicznego hipergrafu z wartościami atrybutów określającymi wymagania projektowe. Jeśli otrzymana wiadomość jest hipergrafem odpowiadającym rozwiązaniu podzadania wygenerowanym przez innego agenta, wówczas generowany hipergraf jest odpowiednio modyfikowany przez agenta.

Jeśli wiadomość jest żądaniem rozwiązania podzadania, atrybuty hipergrafu reprezentującego aktualny wewnętrzny stan agenta są uaktualniane zgodnie z wartościami otrzymanymi w wiadomości. W kolejnym kroku agent rozpoczyna generowanie nowego hipergrafu reprezentującego rozwiązanie. Należy podkreślić, iż agent może być wywołany przez innego agenta za każdym razem z innymi początkowymi wartościami atrybutów. W ten sposób zmiana wartości atrybutów hipergrafów, w których są zakodowane wymagania projektowe, pozwala agentowi zaadaptować się do nowej specyfikacji problemu projektowego.

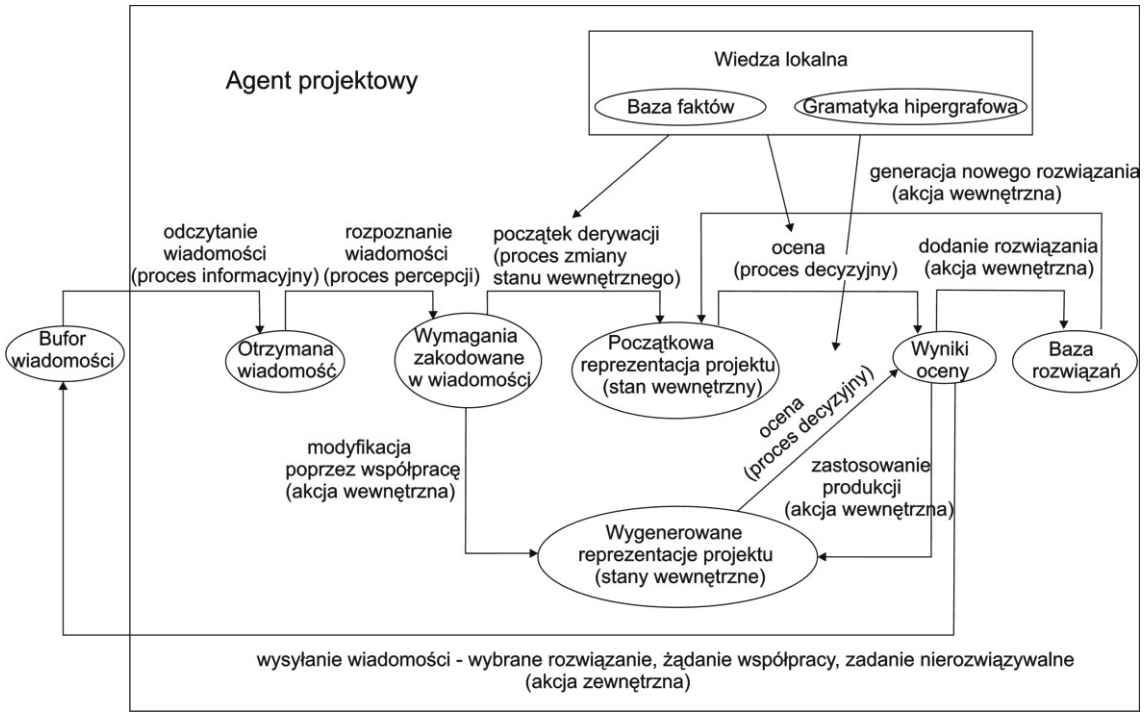
Po każdej modyfikacji generowanego hipergrafu agent ocenia aktualne rozwiązanie i podejmuje decyzję dotyczącą kolejnej akcji. Następnie albo wysyła wiadomość do bufora (wy-

generowane rozwiązanie, żądanie współpracy, powiadomienie o błędzie) albo podejmuje akcję wewnętrzną. W wyniku akcji wewnętrznej wykonuje kolejny krok wyprowadzenia hipergrafu za pomocą wybranej produkcji gramatyki lub zapamiętuje stworzone rozwiązanie i rozpoczyna generację nowego. Dzięki ocenie rozwiązań częściowych reprezentujących aktualne stany projektu agent podejmuje decyzje dotyczące kolejno stosowanych produkcji gramatyki i w ten sposób wybiera kierunek poszukiwań dopuszczalnych rozwiązań w przestrzeni projektowej.

7.2. Reprezentacja wiedzy w systemie wieloagentowym

Agenci, którym są przypisane konkretne zadania projektowe, stanowią moduły systemu wspomagającego projektowanie. Każdy agent jest zdolny do autonomicznych akcji w określonym dla niego środowisku. Jego działanie jest określone poprzez proces percepcji, planowanie i wykonywanie akcji projektowych oraz ocenę aktualnego stanu projektu.

W rozdziale tym zostały przyjęte oznaczenia i koncepcje związane z systemem agentowym, które są modyfikacją pojęć wprowadzonych w [FHMV95, Saun01, Wood99, WoLo00].



Rys. 7.2. Sposób pracy agenta wykorzystującego gramatykę hipergrafową
Fig. 7.2. The way in which a hypergraph grammar-based agent works

Środowisko agenta jest wyznaczone przez nieskończony zbiór stanów $W = \{w_0, w_1, \ldots\}$. Działanie agenta zmieniające środowisko jest scharakteryzowane za pomocą akcji należących do zbioru $A = \{a^0, a^1, \ldots\}$. Każdy agent powinien pamiętać podejmowane decyzje i zaistniałe

stany jego środowiska. Zakładamy, że agent ma dwa typy pamięci: pamięć krótkoterminową i długoterminową. Pamięć krótkoterminowa M_S pozwala agentowi zapamiętywać kilka ostatnich stanów i podjętych decyzji. Stany M_S są scharakteryzowane przez percepcję i formułowanie nowych akcji projektowych za pomocą dwóch zbiorów: zbioru *stanów percepcyjnych* $\mathcal{P} = \{p^0, p^1,\ldots\}$ oraz zbioru *stanów konceptualnych* $\mathcal{C} = \{c^0, c^1,\ldots\}$. Pamięć długoterminowa M_L przechowuje uogólnione doświadczenia z wcześniejszych etapów działania. Każdy stan wewnętrzny inteligentnego agenta jest elementem iloczynu kartezjańskiego obu rodzajów pamięci $Int = M_S \times M_L$.

Definicja 7.1:

Inteligentny agent jest systemem $Ag = (Int, \mathcal{A}, \sigma, \rho, \eta, \chi, \alpha)$, gdzie:

- $Int = \{int^0, int^1,\ldots\}$ jest zbiorem *stanów wewnętrznych* agenta,
- $\mathcal{A} = \{a^0, a^1,\ldots\}$ jest zbiorem *akcji*,
- σ: $\mathcal{W} \rightarrow \mathcal{S}$ definiuje *proces zbierania informacji* o środowisku agenta, gdzie $\mathcal{S} = \{s^0, s^1,\ldots\}$ jest zbiorem *stanów struktury* przechowującej informacje,
- ρ: $\mathcal{S} \rightarrow \mathcal{P}$ definiuje *proces percepcji* będący interpretacją danych ze stanu struktury,
- η: $Int \times \mathcal{P} \rightarrow Int$ definiuje *proces zmiany stanu wewnętrznego* agenta,
- χ: $Int \rightarrow \mathcal{C}$ definiuje *proces decyzyjny*, w którym stan konceptualny jest określany na podstawie stanu wewnętrznego,
- α: $\mathcal{C} \rightarrow \mathcal{A}$ wyznacza akcję podejmowaną przez agenta na podstawie aktualnego stanu konceptualnego. □

W prezentowanym podejściu zbiór zewnętrznych stanów $\mathcal{W} = \{w_0, w_1,\ldots\}$ agentów projektowych jest zbiorem hierarchicznych hipergrafów rozmieszczenia reprezentujących rozwiązania i generowanych równolegle przez wszystkich agentów systemu. Zbiór wewnętrznych stanów agenta *Int* zawiera hierarchiczne hipergrafy wyprowadzane z wykorzystaniem jego gramatyki. Stany percepcyjne odpowiadają wymaganiom otrzymywanym poprzez wiadomości, podczas gdy stany konceptualne reprezentują wyniki oceny generowanych hipergrafów, które są zapamiętywane w zmiennych pamięci krótkoterminowej. Pamięć długoterminowa ma postać bazy danych stworzonych dopuszczalnych rozwiązań. Akcje agenta projektowego są przedstawione na rys. 7.2.

System wieloagentowy *AS* składa się z określonego środowiska *Env* i zbioru działających w nim agentów $Ag_1,\ldots, Ag_n$. Definicja systemu wieloagentowego zostanie poprzedzona definicją środowiska [FHMV95, WoLo00].

Definicja 7.2:

Środowisko jest systemem $Env = (\mathcal{W}, \mathcal{A}_E, K_1,\ldots, K_n, \tau)$, gdzie:

- $\mathcal{W} = \{w_0, w_1,\ldots\}$ jest zbiorem możliwych stanów środowiska,

- $A_E = \{a_E^0, a_E^1, \ldots\}$ jest zbiorem akcji środowiska,
- K_i: $W \to W_{Agi} \in P(W)$ wyznacza środowisko dla agenta Ag_i, czyli jest podziałem środowiska W charakteryzującym informację dostępną dla agenta Ag_i,
- τ: $W \times A_E \times A_1 \times \ldots \times A_n \to P(W)$ wyznacza nowy podzbiór stanów W na podstawie stanu poprzedniego, akcji środowiska i jednej akcji wykonywanej przez każdego z pozostałych agentów. □

Definicja 7.3:

System wieloagentowy jest strukturą $AS = (Env, Ag_1, \ldots, Ag_n)$, gdzie:

- Env jest środowiskiem określonym w definicji 7.2,
- $Ag_1, \ldots, Ag_n$ są inteligentnymi agentami określonymi w definicji 7.1. □

W każdym momencie czasu system znajduje się w pewnym stanie globalnym. Zbiór stanów globalnych GS systemu AS zawiera podzbiory $W \times Int_1 \times \ldots \times Int_n$. Funkcjonowanie systemu jest realizowane poprzez zmiany stanów globalnych opisane funkcją zmiany stanu środowiska τ oraz funkcjami zmiany stanów wewnętrznych agentów η_i.

Początkowy stan globalny ma postać $g_0 = (w_0, int_1^0, \ldots, int_n^0)$. W tym momencie każdy agent Ag_i zbiera informacje o środowisku i określa początkowy stan struktury $s_i^0 = \sigma_i(K_i(w_0))$, na podstawie s_i^0 określa początkowy stan percepcyjny p_i^0, uaktualnia swój stan wewnętrzny obliczając $int_i^1 = \eta_i(int_i^0, p_i^0)$ i wybiera pierwszą akcję, jaka powinna być wykonana $a_i^0 = \alpha_i(\chi_i(int_i^1))$. Stan środowiska jest uaktualniany na $w_1 = \tau(w_0, a_E^0, a_1^0, \ldots, a_n^0)$, a system osiąga kolejny stan globalny $g_1 = (w_1, int_1^1, \ldots, int_n^1)$.

W prezentowanym systemie początkowy stan w_0 ma postać jednej hiperkrawędzi obiektowej wraz z przypisanymi jej wierzchołkami, od której rozpoczyna się proces generacji hipergrafu. Atrybuty tej hiperkrawędzi określają wymagania projektowe, takie jak liczba pokoi domu. Początkowe stany wewnętrzne agentów $int_1^0, \ldots, int_n^0$ są pustymi hipergrafami. Gdy agent Ag_i odkrywa, że jest wywoływany przez środowisko lub innego agenta, utożsamia otrzymaną wiadomość z początkowym hipergrafem (aksjomatem) swojej gramatyki, która staje się jego kolejnym stanem wewnętrznym int_i^1.

Do reprezentowania systemów wieloagentowych i wnioskowania dotyczącego wielu ich aspektów często jest wykorzystywana logika wielomodalna [GaPS90, TrSp95, Wood95]. W celu reprezentacji wiedzy posiadanej przez agentów projektowych zostanie użyty modalny rachunek predykatów. Zostanie zdefiniowany interpretowany system wieloagentowy I_{AS}, a następnie zostanie związana z nim struktura Kripkego [FHMV95].

Załóżmy, że Φ jest zbiorem predykatów opisujących podstawowe fakty dotyczące systemu. Przebieg *AS* nad *GS* jest funkcją r: $\aleph \rightarrow GS$, czyli może być traktowany jako ciąg stanów globalnych z *GS*. Niech R oznacza zbiór przebiegów systemu nad *GS*.

Definicja 7.4:

Interpretowany system wieloagentowy I_{AS} generowany przez strukturę *AS* jest parą (GS_{AS}, π), gdzie:

- $GS_{AS} = \bigcup_{r \in R} r(\aleph)$ jest zbiorem globalnych stanów osiąganych przez *AS*,
- π jest interpretacją predykatów z Φ nad *GS*, która wartościuje predykaty w stanach globalnych. □

Istnieje odpowiedniość jeden do jednego pomiędzy każdym podziałem W, wyznaczającym stany środowiska nierozróżnialne dla agenta Ag_i będącego w określonym stanie z W (definicja 7.2) i relacją równoważności na W. Dla danego podziału K_i zbioru W odpowiadająca mu relacja równoważności K_I jest zdefiniowana w następujący sposób: $(w', w'') \in K_I \Leftrightarrow K_i(w') = K_i(w'')$.

Interpretowany system I_{AS} odpowiada strukturze Kripkego $M_{I_{AS}} = (GS_{AS}, \pi, K_1, \ldots, K_N)$, gdzie $K_I \subseteq GS_{AS} \times GS_{AS}$ jest relacją równoważności odpowiadającą podziałowi K_i zbioru W i rozszerzoną na GS_{AS} w taki sposób, że dla każdych dwóch stanów globalnych $g' = (w', int_1', \ldots, int_n')$ i $g'' = (w'', int_1'', \ldots, int_n'')$, $(g', g'') \in K_I$, jeśli $(w', w'') \in K_I$ i $int_j' = int_j''$.

Niech $\mathcal{K}_1, \ldots, \mathcal{K}_n$ oznaczają operatory modalne, a *L* oznacza język otrzymany przez domknięcie Φ nad negacją, koniunkcją i operatorami $\mathcal{K}_1, \ldots, \mathcal{K}_n$.

Definicja 7.5:

Agent Ag_i wie, że formuła $\varphi \in L$ jest spełniona w stanie globalnym $g = (w, int_1, \ldots, int_n)$ systemu *AS*, jeśli $(M_{I_{AS}}, g) \models \varphi$ (tzn. φ jest spełnione w stanie *g* struktury $M_{I_{AS}}$).

Innymi słowy: $(I_{AS}, g) \models \mathcal{K}_i\varphi \Leftrightarrow \pi(h)(\varphi) = true$ dla wszystkich *h* takich, że $(g, h) \in K_I$. □

Wiedza agenta projektowego mieści się w jego bazie danych, regułach hierarchicznej gramatyki hipergrafowej i generowanych hierarchicznych hipergrafach. Korzystając ze zbioru Φ predykatów opisujących kryteria projektowe, agent może sprawdzić poprawność wygenerowanego rozwiązania poprzez porównanie wiedzy zakodowanej w hipergrafie z danymi ograniczeniami.

7.3. Język hipergrafowy generowany przez agentów projektowych

Każdy agent projektowy posiada programowaną hierarchiczną hipergrafową gramatykę rozmieszczenia, która ma wpływ na jego działanie. Podczas wywołania agent otrzymuje wartościowany hipergraf początkowy, który utożsamia z aksjomatem swojej gramatyki. Wartości atrybutów przypisane aksjomatowi odpowiadają wymaganiom projektowym i od nich zależy, które produkcje gramatyki będą wybierane przez agenta w trakcie wywodu hipergrafu reprezentującego rozwiązanie. Wygenerowane hierarchiczne hipergrafy rozmieszczenia mogą być następnie odwzorowywane w różne modele graficzne projektowanych obiektów, zgodnie z wartościami przypisanych im atrybutów.

Zdefiniujemy teraz język złożony z elementów wygenerowanych przez współpracujących ze sobą agentów, którzy nie tylko równolegle modyfikują różne części wyprowadzanego hipergrafu, ale także zlecają wygenerowanie pewnych części hipergrafu innym agentom. Żądanie współpracy ma miejsce po znalezieniu przez agenta w generowanym przez siebie hipergrafie rozmieszczenia hiperkrawędzi obiektowej, która nie jest hierarchiczna i jest etykietowana za pomocą jednego z elementów wyspecyfikowanego zbioru etykiet zwanych *symbolami komunikacyjnymi* [CDKP94]. Wywołanie innego agenta polega na wysłaniu do niego wiadomości w postaci uchwytu indukowanego przez znalezioną hiperkrawędź. Symbole komunikacyjne identyfikują agentów odpowiedzialnych za rozwiązywanie określonych podzadań. Na przykład, etykiety *House Interior* i *Garden* wskazują, że podzadania zostaną zlecone agentom projektującym odpowiednio wnętrze domu i ogród, a etykiety ze zbioru {*livingroom*, *room*, *bathroom*, *kitchen*} – że zostaną wywołani agenci aranżujący określone rodzaje pomieszczeń. Wspólna generacja hipergrafu rozmieszczenia przez agentów projektowych odbywa się w sposób analogiczny jak generacja hipergrafu w rozproszonym systemie projektowania wykorzystującym system gramatyk hipergrafowych [GSŚG08, GrSŚ09]. Zastosowane tam gramatyki nie są jednak gramatykami hierarchicznymi.

Niech $\mathcal{G}_{P1},\ldots,\mathcal{G}_{Pn}$ oznaczają bezkontekstowe programowane hierarchiczne gramatyki hipergrafowe agentów projektowych. Generacja hipergrafu reprezentującego rozwiązanie rozpoczyna się od aksjomatu $\boldsymbol{X}_1$ gramatyki agenta-menadżera $\mathcal{G}_{P1}$ i w pierwszym kroku zostaje zastosowana jedna z produkcji tej gramatyki. Następnie agent-menadżer może wywołać innych agentów i proces wywodu odbywa się w sposób równoległy. Agent Ag_i, otrzymujący wiadomość w postaci hipergrafu zawierającego tylko jedną niehierarchiczną hiperkrawędź obiektową h, utożsamia h z aksjomatem $\boldsymbol{X}_i$ swojej gramatyki i rozpoczyna lokalny proces wywodu hipergrafu. W każdym kroku agent Ag_i albo wykonuje przepisywanie na lokalnie wyprowadzanym hipergrafie za pomocą jednej z produkcji gramatyki $\mathcal{G}_{Pi}$ albo zastępuje uchwyt w tym hipergrafie hierarchicznym hipergrafem wygenerowanym przez wcześniej

wywołanego przez niego agenta. Po znalezieniu hiperkrawędzi obiektowych etykietowanych za pomocą symboli komunikacyjnych agent Ag_i wywołuje kolejnych agentów.

Niech $\mathcal{H}$ oznacza rodzinę atrybutowanych hierarchicznych hipergrafów rozmieszczenia nad Σ i A.

Definicja 7.6:
Niech $h, h', h'', g \in \mathcal{H}$.

Proces wyprowadzenia hierarchicznego hipergrafu rozmieszczenia w systemie wieloagentowym z agentami wyposażonymi w programowane hierarchiczne hipergrafowe gramatyki rozmieszczenia $\mathcal{G}_{P1},..., \mathcal{G}_{Pn}$ składa się z bezpośrednich wyprowadzeń dwóch następujących postaci:

1. h'' jest **bezpośrednio wyprowadzone** z h' ($h' \Rightarrow_1 h''$) z wykorzystaniem produkcji p gramatyki $\mathcal{G}_{Pi}$, $1 \le i \le n$, zgodnie z definicją 5.3, i przechodząc od wierzchołka v do wierzchołka w o etykiecie p w diagramie sterującym gramatyki $\mathcal{G}_{Pi}$,
2. h'' jest **bezpośrednio wyprowadzone** z h' ($h' \Rightarrow_2 h''$), jeśli istnieje w h' uchwyt h indukowany przez niehierarchiczną hiperkrawędź obiektową i izomorficzny z aksjomatem X_i gramatyki $\mathcal{G}_{Pi}$, $1 \le i \le n$, g jest atrybutowanym hierarchicznym hipergrafem rozmieszczenia wygenerowanym z X_i w $\mathcal{G}_{Pi}$ oraz h'' jest izomorficzny z wynikiem zastąpienia h w h' przez g i zastąpienia zewnętrznych wierzchołków h odpowiadającymi im zewnętrznymi wierzchołkami g. □

Język generowany przez wieloagentowy system projektowy jest zbiorem hierarchicznych hipergrafów rozmieszczenia, które zostały wygenerowane przez różnych agentów, zaczynając od aksjomatu X_1 gramatyki $\mathcal{G}_{P1}$ agenta-menadżera.

Definicja 7.7:
Niech *DAS* będzie wieloagentowym systemem projektowym z agentami wyposażonymi w programowane hierarchiczne hipergrafowe gramatyki rozmieszczenia $\mathcal{G}_{P1},..., \mathcal{G}_{Pn}$. Niech v_I, v i v_F będą wierzchołkami diagramu sterującego gramatyki $\mathcal{G}_{P1}$, gdzie v_I oraz v_F oznaczają początkowy i końcowy wierzchołek tego diagramu.

Język generowany przez *DAS* jest zbiorem:
$L(DAS) = \{h \in \mathcal{H} \mid (X_1, v_I) \Rightarrow_1 (h', v) \{\Rightarrow_1, \Rightarrow_2\}^* (h, v_F)\}$. □

7.4. Semantyczny model wieloagentowego systemu projektowego wykorzystującego gramatyki hipergrafowe

Każdy agent *Ag* systemu projektowego jest wyposażony w programowaną hierarchiczną hipergrafową gramatykę rozmieszczenia $\mathcal{G}_P$, która generuje hipergrafowe reprezentacje rozwiązań projektowych, oraz bazę *B* zawierającą wiedzę dotyczącą podzadania, za które agent jest odpowiedzialny. Na przykład, agent projektujący wnętrze domu posiada w swojej bazie wiedzy zbiór norm architektonicznych, takich jak minimalne powierzchnie pokoi i ich pożądana lokalizacja geograficzna.

Niech para $LK = (\mathcal{G}_P, B)$ reprezentuje lokalną wiedzę projektową agenta.

Definicja 7.8:

Agent projektowy $DA = (Ag, LK)$ jest inteligentnym agentem $Ag = (Int, A, \sigma, \rho, \eta, \chi, \alpha)$ wyposażonym w lokalną wiedzę projektową zorientowaną problemowo $LK = (\mathcal{G}_P, B)$. □

Proces projektowy odbywa się w określonym kontekście zmieniającym się w czasie. Kontekst ten jest wyrażony przez środowisko, w którym działają agenci, i predykaty opisujące kryteria projektowe. Agenci dostarczając sobie informacji także stanowią wzajemnie dla siebie kontekst. Agent żądający współpracy wywołuje innych agentów z określonymi wymaganiami projektowymi zakodowanymi w atrybutach uchwytu hipergrafu stanowiącego wiadomość. Wartości tych atrybutów zmieniają się wraz z nowymi predykatami.

Niech *GS* oznacza zbiór globalnych stanów wieloagentowego systemu projektowego.

Definicja 7.9:

Kontekst wieloagentowego systemu projektowego jest strukturą $\gamma = (Env, g_0, \psi)$, gdzie:

- *Env* jest środowiskiem systemu,
- $g_0 \in GS$ jest początkowym stanem globalnym,
- ψ jest zbiorem predykatów opisujących wymagania i ograniczenia projektowe. □

Zbiór agentów projektowych, określony kontekst i bufor komunikacyjny tworzą razem wieloagentowy system projektowy**.**

Definicja 7.10:

Wieloagentowy system projektowy jest strukturą $DAS = (\gamma, M_B, DA_1, \ldots, DA_n)$, gdzie:

- γ jest kontekstem wieloagentowego systemu projektowego,
- M_B jest buforem komunikacyjnym wykorzystywanym przez agentów i środowisko,
- $DA_1, \ldots, DA_n$ są agentami projektowymi. □

Jak już było wspomniane, stany środowiska agentów projektowych są atrybutowanymi hierarchicznymi hipergrafami rozmieszczenia generowanymi wspólnie przez agentów systemu. Wieloagentowy system projektowy działa w następujący sposób. Będąc w początkowym globalnym stanie środowisko wykonuje akcję $send(h_0, 1, 0)$, gdzie h_0 jest wysyłaną wiadomo-

ścią, a 0 i 1 oznaczają odpowiednio środowisko i agenta-menadżera. h_0 jest hipergrafem zawierającym jedną niehierarchiczną hiperkrawędź obiektową i przypisane jej wierzchołki. Wszystkie atomy h_0 posiadają poprawnie określone wartości atrybutów. W kolejnym kroku agent-menadżer rozpoczyna proces wyprowadzenia hipergrafu poprzez utożsamienie h_0 z aksjomatem swojej gramatyki. W następnych krokach stosuje on odpowiednie reguły gramatyki lub zleca dalsze wyprowadzenie innym agentom poprzez wysłanie do nich wiadomości.

Zbiór *Int* wewnętrznych stanów agenta projektowego jest zbiorem hipergrafów wyprowadzonych z wykorzystaniem produkcji należących do jego gramatyki hipergrafowej bądź poprzez zastąpienie w trakcie wywodu pewnych uchwytów hipergrafami wygenerowanymi przez współpracujących z nim agentów. Każdy stan wewnętrzny int^k odpowiada hipergrafowi wygenerowanemu po k krokach wyprowadzenia. Zbiór akcji agenta $A = A_I \cup A_E$ zawiera akcje wewnętrzne A_I i akcje postaci $a^i = send(\mu, j, n) \in A_E$, gdzie μ oznacza wysyłaną wiadomość, j jest identyfikatorem agenta, do którego wiadomość jest wysyłana, a n jest identyfikatorem agenta, który wysyła wiadomość. Każda wewnętrzna akcja $a^i \in A_I$ zmienia aktualny stan wewnętrzny agenta poprzez zastosowanie reguły gramatyki hipergrafowej do generowanego hipergrafu, zastąpienie hiperkrawędzi tego hipergrafu hipergrafem otrzymanym od innego agenta lub przez zapamiętanie wygenerowanego rozwiązania i rozpoczęcie generacji nowego. Dodatkowo, zbiór akcji zawiera specjalną akcję *null*, która mówi, że agent w danym kroku niczego nie wykonuje.

W procesie zbierania informacji σ agent projektowy szuka w buforze adresowanych do niego wiadomości. Tak więc jeśli agent DA_n wykonał akcję $send(\mu, j, n)$, wówczas agent DA_j, w k-tym kroku w wyniku testowania bufora otrzyma wiadomość μ, której rodzaj jest określany w procesie percepcji ρ. W kolejnym kroku, w procesie zmiany wewnętrznego stanu agenta wartości zmiennych projektowych są uaktualniane zgodnie z typem otrzymanej wiadomości. Wiadomość μ jest zawsze atrybutowanym hipergrafem rozmieszczenia, który może mieć postać:

- pustego hipergrafu – stanowi powiadomienie o błędzie. Wiadomość ta oznacza, że agent DA_n nie był w stanie rozwiązać zleconego mu podzadania. Wówczas hiperkrawędź w hipergrafie generowanym przez agenta DA_j, która miała zostać rozwinięta przez agenta DA_n pozostanie niezmieniona;
- hipergrafu zawierającego więcej niż jedną niehierarchiczną hiperkrawędź obiektową – oznacza, że agent DA_n zwrócił hipergraf odpowiadający rozwiązaniu podzadania. W takim przypadku odpowiedni uchwyt w hipergrafie generowanym przez agenta DA_j jest zastępowany przez μ. Jeśli środowisko otrzymuje wiadomość takiej postaci wówczas reprezentuje ona kompletne rozwiązanie danego zadania projektowego, które jest dodawane do globalnej bazy danych rozwiązań;

- hipergrafu złożonego z jednej niehierarchicznej hiperkrawędzi obiektowej z przypisanymi jej wierzchołkami – oznacza, że agent DA_n zlecił wykonanie podzadania agentowi DA_j. Wówczas wartości zmiennych z pamięci krótkoterminowej agenta DA_j są uaktualniane zgodnie z wartościami atrybutów hipergrafu μ, μ jest utożsamiane z aksjomatem gramatyki $\mathcal{G}_{PPj}$ i jest rozpoczynana generacja nowego hierarchicznego hipergrafu rozmieszczenia.

Po każdym kroku wywodu hipergrafu agent ocenia swój aktualny stan wewnętrzny reprezentujący rozwiązanie częściowe i wybiera kolejną akcję w procesie podejmowania decyzji χ. Jeśli nie da się już zastosować żadnej reguły gramatyki, a wygenerowany hipergraf h nie należy do języka generowanego przez agenta DA_j (agent w procesie wywodu nie doszedł do wierzchołka końcowego diagramu sterującego gramatyki), wiadomość w postaci pustego hipergrafu jest przesyłana do agenta DA_n, który wywołał agenta DA_j. Jeśli h jest hipergrafem z języka generowanego przez agenta DA_j wraz z wywoływanymi przez niego agentami, wiadomość $\mu = h$ jest wysyłana do agenta DA_n. Jeśli jest potrzebna współpraca innego agenta, wówczas do bufora jest wysyłana wiadomość w postaci uchwytu o pustej zawartości, który powinien zostać rozwinięty przez innego agenta.

Zdolność agenta do oceny rozwiązania zależy od posiadanej przez niego wiedzy. Ocena ta bazuje na wiedzy projektowej zawartej zarówno w danych dotyczących podzadania, za które agent jest odpowiedzialny, predykatach opisujących wymagania i ograniczenia projektowe wyspecyfikowane w kontekście projektowym, jak i w strukturze generowanego hipergrafu oraz wartościach atrybutów przypisanych jego atomom. Agent ocenia zarówno częściowe, jak i kompletne rozwiązania. Ocena rozwiązań częściowych ma wpływ na podejmowane przez agenta decyzje, dotyczące wymaganych modyfikacji i wyboru kierunku dalszych poszukiwań rozwiązania.

Każdy agent projektowy posiada konceptualizację zadaniową dotyczącą dziedziny rozwiązywanego przez niego podzadania. Na przykład, w konceptualizacji agenta projektującego rozkłady pomieszczeń obiektami są pomieszczenia, obszary i ściany, a relacje określają dostępność i przyległość między obiektami, w konceptualizacji agenta aranżującego pomieszczenia obiektami są pomieszczenia, meble i sprzęty, a relacje określają odległości między tymi obiektami, w konceptualizacji agenta projektującego ogród obiektami są rośliny, trawniki, grządki, ścieżki, oczka wodne i meble ogrodowe, a relacje określają wzajemne położenie tych obiektów. Każdy agent zna też konceptualizacje zadaniowe wykorzystywane przez wywoływanych przez siebie agentów, dzięki czemu potrafi interpretować hipergrafy rozmieszczenia wygenerowane wspólnie z innymi agentami.

W rozważanym ontologicznym podejściu do modelowania systemu agentowego relacje pomiędzy obiektami zdefiniowane w konceptualizacji projektowej mogą zachodzić we wszystkich możliwych stanach środowiska ze zbioru W. Wykorzystanie logiki do opisu wie-

dzy dotyczącej kolejnych stanów osiąganych przez system w wyniku akcji wykonywanych przez agentów wymaga ontologicznego dopasowania słownika formuł logicznych do zbioru konceptualizacji zadaniowych agentów w sposób analogiczny jak w definicji 6.2. Następnie są definiowane predykaty wyrażające zarówno wiedzę o projektach zawartą w hipergrafach, jak i kryteria projektowe dla podzadań rozwiązywanych przez agentów. Mając interpretację predykatów określoną przez strukturę relacyjną przypisującą ich elementom składowym atomy hipergrafów i ich atrybuty, agenci oceniają zgodność generowanych przez siebie hipergrafów względem formuł z języka otrzymanego przez domknięcie zbioru predykatów nad negacją, koniunkcją i operatorami modalnymi. Agent wie, że dana formuła jest spełniona przez wygenerowany hipergraf, jeśli jest ona prawdziwa przy podanej interpretacji dla wszystkich izomorficznych z nim hipergrafów. Agent-menadżer, który scala rozwiązania otrzymywane od innych agentów, zna konceptualizacje charakteryzujące wiedzę projektową wszystkich agentów. Pozwala mu to ocenić zgodność wygenerowanego wspólnie rozwiązania z zadanymi ograniczeniami i wymaganiami projektowymi.

Niech $\mathcal{C} = (O, O_R, A, Att, Z, R)$ oznacza konceptualizację będącą zbiorem konceptualizacji zadaniowych wszystkich agentów projektowych systemu. Niech $\mathcal{V} = \{\mathcal{S}, \mathcal{F}, \mathcal{P}\}$ oznacza słownik formuł złożony ze zbioru rodzajów, zbioru symboli *n*-arnych wielorodzajowych funkcji ze zbiorem symboli stałych oraz zbioru symboli *n*-arnych wielorodzajowych predykatów. Dopasowanie ontologiczne między $\mathcal{V}$ i $\mathcal{C}$ przypisuje symbolom stałych ze zbioru O obiekty aktywne ze zbioru O_R, symbolom funkcji o wartościach ze zbioru D – atrybuty ze zbioru A, a symbolom predykatów o argumentach ze zbioru O – relacje ze zbioru R. Zbiór termów jest tworzony przez domknięcie symboli stałych nad zastosowaniem funkcji.

Zbiór ψ składa się z predykatów zdefiniowanych nad $\mathcal{V}$ postaci $P(t_1,\ldots, t_n)$ i $t_1 = t_2$, gdzie $t_1,\ldots, t_n$ są termami tego samego rodzaju, a P jest symbolem *n*-arnego predykatu. Niech L oznacza język formuł otrzymany przez domknięcie ψ nad negacją, koniunkcją i operatorami modalnymi $\mathcal{K}_1,\ldots, \mathcal{K}_n$. Niech DB oznacza globalną bazę danych systemu zawierającą hierarchiczne hipergrafy rozmieszczenia należące do języka $L(DAS)$ generowanego przez wieloagentowy system projektowy DAS.

Definicja 7.11:
Niech π będzie interpretacją dla formuł z L nad $L(DAS)$ wyznaczoną przez $\mathcal{V}$ – strukturę relacyjną z dziedziną złożoną z hiperkrawędzi obiektowych i wierzchołków hipergrafów z $L(DAS)$ oraz wartości przypisanych im atrybutów.

Interpretowany system projektowy I_{DAS} generowany przez strukturę DAS jest parą (DB, π). □

Niech $g' \subseteq g \in DB$ będzie hipergrafem rozmieszczenia wygenerowanym przez agenta projektowego DA_i i stanowiącym podgraf hipergrafu g.

Definicja 7.12:

Agent DA_i wie, że formuła $\varphi \in L$ jest spełniona dla $g \supseteq g'$ w systemie I_{DAS}, $(I_{DAS}, g) \models \mathcal{K}_i\varphi \Leftrightarrow \pi(\varphi(h)) = true$ dla wszystkich $h \in DB$ zawierających podgraf izomorficzny z g'. □

Stwierdzenie 7.1:

Dany jest interpretowany system projektowy $I_{DAS} = (DB, \pi)$.

Hipergraf $g \in DB$ jest **poprawnym rozwiązaniem projektowym** względem $\varphi \in L$, jeśli agent-menadżer DA_M, który rozpoczyna generację i łączy wygenerowane rozwiązania, wie, że formuła φ jest spełniona dla g. Innymi słowy:

$(I_{DAS}, g) \models \mathcal{K}_M\varphi \Leftrightarrow \pi(\varphi(h)) = true$ dla wszystkich $h \in DB$ izomorficznych z g. ♦

Stwierdzenie 7.2:

Interpretowany system projektowy I_{DAS} generuje **rozwiązania projektowe zgodne z kryteriami projektowymi**, jeśli $\forall\, g \in DB\ \forall\ \varphi\ \in L\ (I_{DAS}, g) \models \mathcal{K}_M\varphi$ ♦

Każdy agent projektowy DA_i posiada w swojej bazie wiedzy B_i realizację I_i określoną dla hierarchicznych hipergrafów rozmieszczenia z generowanego przez siebie języka. Realizacja ta określa semantyczne znaczenie atomów danego hipergrafu i pozwala agentom w pożądany sposób rozmieścić i dopasować odpowiadające tym atomom komponenty obiektu w celu stworzenia jego wizualizacji. Realizacja I dla DAS jest zdefiniowana jako $I = I_1 \cup \ldots \cup I_n$, gdzie $I_1, \ldots, I_n$ oznaczają odpowiednio realizacje zdefiniowane dla języków generowanych przez agentów projektowych $DA_1, \ldots, DA_n$. Dla hipergrafu $h \in L(DAS)$, $I(h)$ jest nazywane modelem graficznym h.

Niech DB zawiera tylko rozwiązania projektowe zgodne z zadanymi kryteriami projektowymi.

Definicja 7.13:

Język wizualny $\mathcal{J}$ tworzony przez interpretowany system projektowy I_{DAS} jest zbiorem modeli graficznych odpowiadających hierarchicznym hipergrafom z bazy DB, $\mathcal{J} = \sum_{h \in DB} I(h)$. □

Jeśli I odwzorowuje hierarchiczne hipergrafy rozmieszczenia w dziedzinę diagramów projektowych, to interpretowany agentowy system projektowy tworzy zbiór diagramów projektowych będący podzbiorem języka wizualnego wykorzystywanego przez użytkownika do komunikacji z systemem wspomagania projektowania wizualnego. Otrzymane diagramy stanowią potencjalne rozwiązania proponowane projektantowi przez agentów, które mogą być w dalszej pracy nad projektem przekształcane za pomocą akcji projektowych.

7.5. Kreatywne projektowanie z wykorzystaniem inteligentnych agentów projektowych

Na początku procesu projektowego wszystkie informacje dotyczące danego zadania projektowego są dostarczane przez projektanta, który powinien uwzględnić wszelkie wymagania klienta. W rozważanym przykładzie informacja wejściowa obejmuje opis posiadłości, ogrodu, garażu i pokoi, jakie są pożądane w domu, jak również styl i typ mebli. Globalna baza danych wygenerowanych rozwiązań może być początkowo pusta lub może zawierać pewne prototypowe rozwiązania podobnych zadań, natomiast bufor komunikacyjny, przez który agenci komunikują się ze sobą, jest pusty.

7.5.1. Agent-menadżer projektu

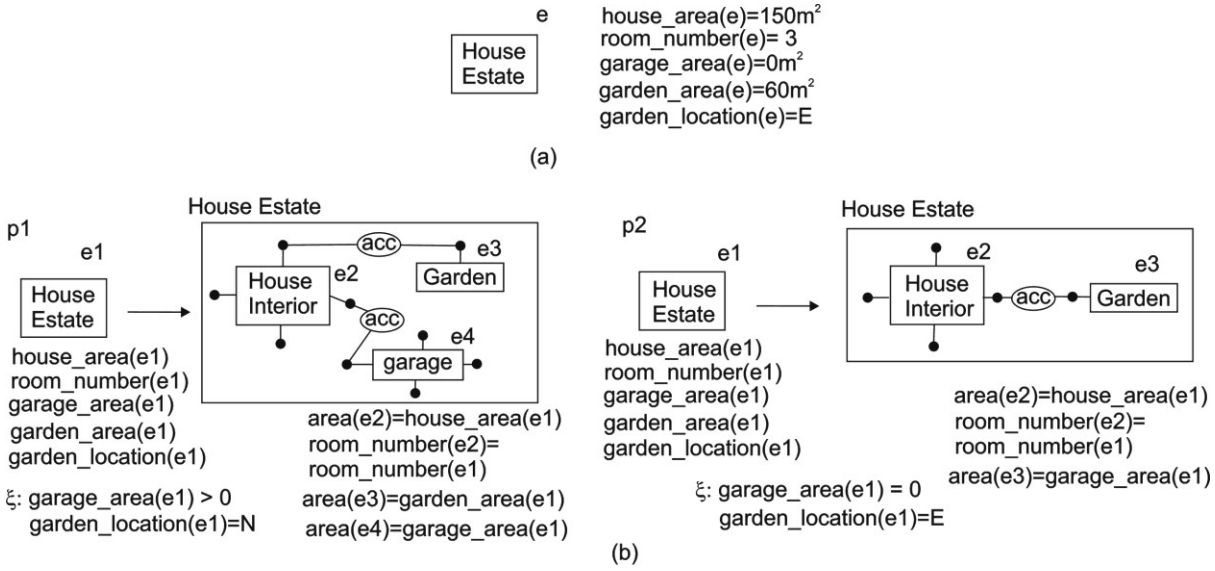
Wymagania i ograniczenia projektowe są określone w początkowym stanie środowiska w postaci wartości atrybutów początkowego hipergrafu. Agent-menadżer, który jest odpowiedzialny za całe zadanie projektowe, generuje układ posiadłości, wykorzystując reguły swojej hierarchicznej gramatyki hipergrafowej, łączy go z zaaranżowanym rozkładem pomieszczeń domu i projektem ogrodu, które są generowane przez innych agentów i zwraca całe rozwiązanie agentowi-środowisku.

Na początku procesu projektowego środowisko wysyła wiadomość w postaci atrybutowanego hipergrafu do agenta-menadżera. W procesie percepcji agent-menadżer zapamiętuje wartości atrybutów tego hipergrafu opisujące wymagania projektowe, takie jak liczba pokoi w domu, ich kształt i powierzchnia, powierzchnia i cechy ogrodu, potrzeba umieszczenia garażu. Początkowy stan agenta zmienia się poprzez uaktualnienie wartości zmiennych projektowych (początkowa zawartość pamięci krótkoterminowej agenta) i utożsamienie otrzymanego hipergrafu z aksjomatem gramatyki agenta. Będąc w nowym stanie wewnętrznym, agent rozpoczyna wyprowadzanie hierarchicznego hipergrafu rozmieszczenia z wykorzystaniem reguł swojej gramatyki.

Przykład 7.1: rozkład posiadłości

Dwie wybrane produkcje hierarchicznej gramatyki hipergrafowej agenta-menadżera wraz z predykatami ξ określającymi warunki ich stosowania są przedstawione na rys. 7.3b. Pokazane są także wybrane reguły semantyczne definiujące sposób przekazywania wartości atrybutów przypisanych atomom hipergrafów lewej strony produkcji atomom hipegrafów prawej strony. Pierwsza produkcja jest stosowana, jeśli dom powinien mieć garaż, a ogród powinien znajdować się po północnej stronie domu. Druga produkcja pozwala agentowi wygenerować dom bez garażu z ogrodem znajdującym się po wschodniej stronie domu. Hiperkrawędzie etykietowane *House Interior* i *Garden* wskazują, że projektowanie odpowiadających im komponentów zostanie zlecone innym agentom. ◊

Na podstawie swojego aktualnego stanu wewnętrznego agent-menadżer albo wybiera lokalną akcję, albo wysyła wiadomość do innego agenta. Akcja lokalna polega na zastosowaniu reguły gramatyki określonej na podstawie aktualnych wartości zmiennych projektowych i kolejności wyznaczonej przez diagram sterujący gramatyki.



Rys. 7.3. Hierarchiczna gramatyka hipergrafowa: a) hiperkrawędź z wymaganiami projektowymi, b) dwie reguły gramatyki agenta-menadżera

Fig. 7.3. A hierarchical hypergraph grammar: a) a hyperedge with design requirements, b) two rules of a grammar of the manager agent

Przykład 7.2: rozkład posiadłości

Załóżmy, że agent-menadżer otrzymuje wiadomość w postaci początkowej hiperkrawędzi obiektowej pokazanej na rys. 7.3a z ustalonymi wartościami atrybutów, które odzwierciedlają wymagania dotyczące powierzchni domu, ogrodu i garażu, liczby sypialni i lokalizacji ogrodu. Mając taki zbiór przykładowych ograniczeń, agent poszukuje spełniającego je projektu. Wartości atrybutów otrzymane od środowiska skłaniają agenta do wykorzystania na początku reguły drugiej. ◊

Jeśli generowane rozwiązanie nie spełnia wymaganych warunków, agent wysyła wiadomość o błędzie w postaci pustego hipergrafu do środowiska. Jeśli agent znajdzie w generowanym hipergrafie hiperkrawędź z etykietą *House Interior* lub *Garden*, która wskazuje, że jest niezbędna współpraca innego agenta, wysyła on wiadomość w postaci uchwytu indukowanego przez tę hiperkrawędź z aktualnymi wartościami atrybutów do agenta, który powinien wygenerować strukturę miejsca odpowiadającego znalezionej etykiecie. Przeszukując bufor, agent może łatwo znaleźć rozwiązanie zwrócone przez wcześniej wywołanego agenta. Wówczas podejmuje kolejną akcję lokalną polegającą na połączeniu zwróconego hipergrafu z generowanym przez siebie rozwiązaniem poprzez zastąpienie wysłanego wcześniej uchwytu otrzymanym hipergrafem.

7.5.2. Agent projektujący wnętrze domu

Agent projektujący wnętrze domu generuje struktury rozkładów pomieszczeń. Rozpoczyna on generację hipergrafu po otrzymaniu wiadomości w postaci hipergrafu z hiperkrawędzią etykietowaną *House Interior*. Po każdym kroku wyprowadzenia hipergrafu agent ten ocenia rozwiązania częściowe możliwe do otrzymania poprzez zastosowanie dopuszczalnych reguł gramatyki względem informacji zamieszczonych w bazie danych i wymagań projektowych określonych w kontekście projektowym. Jeżeli zastosowanie wybranej reguły powoduje, że generowany hipergraf nie spełnia kryteriów projektowych, wówczas agent próbuje zastosować inną spośród reguł dopuszczalnych przez diagram sterowania na aktualnym etapie wywodu.

Przykład 7.3: rozkłady pomieszczeń
Wybrane reguły hierarchicznej gramatyki hipergrafowej agenta generującego rozkłady pomieszczeń są przedstawione na rys. 5.2. Załóżmy, że agent otrzymał hipergraf z hiperkrawędzią e etykietowaną *House Interior* z wartościami atrybutów $area(e) = 150$ m^2, $room\_number(e) = 3$. Wartości te są nadawane atrybutom aksjomatu gramatyki, który jest izomorficzny z lewą stroną pierwszej produkcji. Po zastosowaniu tej produkcji wartość atrybutu *room_number* hiperkrawędzi etykietowanej *Sleeping*1 jest równa 1 i dlatego w kolejnym kroku wywodu będzie zastosowana produkcja druga. Hipergraf wygenerowany zgodnie z tymi kryteriami jest pokazany na rys. 5.3.

Gramatyka zawiera też produkcję *p*3', która generuje trzy pokoje w części sypialnej *Sleeping*1. Obie produkcje, *p*3 i *p*3', mogą być zastosowane, jeśli wartość atrybutu *room_number* jest większa niż 1. W takim przypadku agent wybiera produkcję, która ma być zastosowana, na podstawie aktualnej wartości atrybutu *area* hiperkrawędzi *Sleeping*1. Jeśli wartość ta jest mniejsza niż 30 m^2, agent tworzy tylko dwa pokoje stosując produkcję *p*3. Po zastosowaniu tej produkcji agent ocenia otrzymany hipergraf rozmieszczenia z wykorzystaniem informacji ze swojej bazy danych. Sprawdza on, czy wartości atrybutu *area* obu hiperkrawędzi etykietowanych *room* mieszczą się w zakresie określonym dla powierzchni sypialni w bazie danych. Jeśli są one mniejsze niż wymagane normy, wiadomość o błędzie jest wysyłana do agenta-menadżera. ◊

W trakcie wywodu hipergrafu rozmieszczenia agent może też wywołać innego agenta. Jeśli generowany hipergraf zawiera hiperkrawędzie etykietowane elementami zbioru {*livingroom*, *room*, *bathroom*, *kitchen*}, agent projektujący wnętrze domu zleca podzadania agentom generującym układy mebli dla określonych rodzajów pomieszczeń. Rozkład pomieszczeń połączony z układem sprzętów w kuchni (lodówka, kuchenka, zlew, zmywarka, lada, taboret, dwie szafki i stół z czterema krzesłami) będący realizacją hierarchicznego hipergrafu

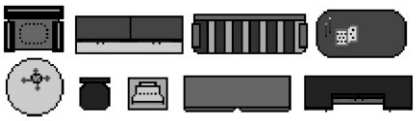
wygenerowanego przez agenta projektującego wnętrze domu przy współpracy agenta projektującego kuchnię jest pokazany na rys. 5.4b.

7.5.3. Agenci projektujący rozkłady mebli

W omawianym wieloagentowym systemie projektowym istnieją czterej agenci generujący rozkłady mebli odpowiednio dla salonu, sypialni, kuchni i łazienki. Każdy z nich jest wywoływany przez agenta projektującego wnętrze domu, jeśli w generowanym przez niego hipergrafie pojawi się odpowiednia etykieta. Bazy danych określone dla tych agentów zawierają zbiory obiektów reprezentowanych graficznie za pomocą ikon odpowiadających meblom i sprzętom. Wartości atrybutów przypisanych tym obiektom definiują kształt, kolor oraz lokalizację sprzętów i mebli.

Przykład 7.4: aranżacja pomieszczeń
Graficzne reprezentacje zbioru mebli zawarte w bazie danych agenta generującego rozkłady mebli w salonie są pokazane na rys. 7.4. Przedstawiają one fotel, komodę, kanapę, stół, stolik, krzesło, telewizor, szafę i regał. Dla szafy i regału wartość atrybutu *location* jest równa 1, co oznacza, że meble te muszą być umieszczone pod ścianą. Dla pozostałych mebli wartość atrybutu *location* wynosi 0, co oznacza, że nie ma ograniczeń co do ich umiejscowienia.

◊



Rys. 7.4. Meble mogące pojawić się w salonie
Fig. 7.4. Furniture of a living-room

Kiedy agent generujący aranżację pomieszczenia znajdzie w buforze komunikacyjnym wiadomość adresowaną do siebie, w procesie percepcji odczytuje atrybuty opisujące kryteria projektowe (jak powierzchnia pomieszczenia, lokalizacja drzwi i okien). Następnie rozpoczyna generację rozmieszczenia mebli z wykorzystaniem odpowiedniej hierarchicznej gramatyki hipergrafowej i bazy wiedzy, gdzie są zawarte informacje dotyczące mebli i sprzętów dla tego pomieszczenia.

Stosując kolejną produkcję podczas generacji hierarchicznego hipergrafu rozmieszczenia, agent projektowy zna wartości atrybutów przypisanych elementom lewej strony produkcji specyfikujących aktualne ograniczenia projektowe, takie jak dostępna powierzchnia pomieszczenia, położenie ścian wraz z drzwiami i oknami, lokalizacja pionów wodnych, kanalizacyjnych i instalacji gazowej. W zależności od tych atrybutów stara się on przypisać wartości atrybutów hiperkrawędzi obiektowych i wierzchołków wstawianego hipergrafu w taki sposób, aby wygenerowany hipergraf rozmieszczenia spełniał wymagania projektowe. Nadaje on wartości atrybutom określającym położenie sprzętów tak, aby wyposażenie, takie jak

wanna, prysznic, umywalka, sedes, zlew, miało dostęp do instalacji wodnej i kanalizacyjnej, a kuchenka i piec gazowy – do instalacji gazowej. Położenie pozostałych mebli i sprzętów zostaje określone w sposób uwzględniający relacje między nimi zdefiniowane przez hiperkrawędzie relacyjne i dotyczące ich wzajemnej odległości (niektóre meble lub grupy mebli powinny być umieszczone z dala od siebie, a inne blisko siebie), ograniczenia zdefiniowane przez atrybut *location* (wartości 0, 1 i 2 oznaczają odpowiednio, że brak ograniczeń co do ich umiejscowienia, że meble te muszą być umieszczone pod ścianą, oraz że meble mogą znaleźć się pod oknem), oraz tak aby nie zachodziły one na siebie. Istnieje także możliwość rozmieszczenia grup mebli wokół zdefiniowanego obiektu centralnego (np. stół i krzesła) lub równolegle frontem do siebie (np. sofa, fotele i telewizor lub kominek w salonie). Po zdefiniowaniu położenia mebli odpowiednio jest modyfikowana wartość atrybutu określającego aktualnie dostępną powierzchnię aranżowanego pomieszczenia.

Analogicznie jak agent-menadżer i agent projektujący wnętrze domu, także agent aranżujący pomieszczenie po każdym kroku wyprowadzenia hipergrafu ocenia możliwe do otrzymania rozwiązania częściowe względem danych wymagań i podejmuje decyzję o kolejnej akcji, jaka powinna zostać podjęta.

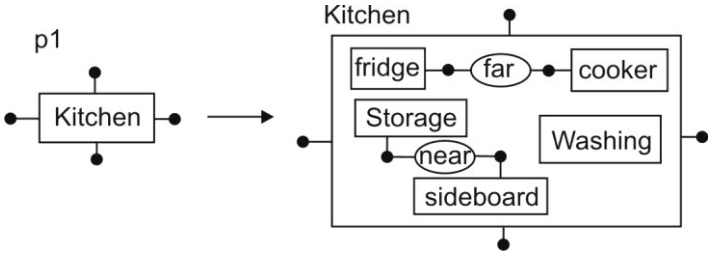
Przykład 7.5: aranżacja pomieszczeń

Wybrane produkcje bezkontekstowej hierarchicznej hipergrafowej gramatyki rozmieszczenia generującej struktury reprezentujące rozkład sprzętów w kuchni i wykorzystywanej przez agenta zajmującego się aranżacją kuchni są przedstawione na rys. 5.4. Diagram sterujący dla tej gramatyki jest pokazany na rys. 5.6b Baza wiedzy tego agenta zawiera informacje o kształcie i kolorze, a także wymaganym sposobie lokalizacji zlewu, zmywarki, kuchenki, lodówki, szafki, lady, taboretu, stołu i krzesła.

Agent stosując pierwszą produkcję, która jest pokazana na rys. 7.5, zna wartości atrybutów określających wolną powierzchnię kuchni (atrybut hiperkrawędzi o etykiecie *Kitchen*), lokalizację ścian wraz z umiejscowieniem drzwi i okien, a także pionów wodnych, kanalizacyjnych i instalacji gazowej (atrybuty wierzchołków reprezentujących ściany kuchni). Rozmieszczanie kolejnych sprzętów w kuchni rozpoczyna od tych, które wymagają odpowiednich instalacji, a więc umieszcza kuchenkę gazową przy ścianie posiadającej przyłącze gazu, a obszar, który będzie zawierał zlew w pobliżu pionu wodnego. Wówczas wartości odpowiednich atrybutów hiperkrawędzi etykietowanych *cooker* i *washing* to *gaz_supply*(*cooker*) = 1 i *water_supply*(*washing*) = 1. Następnie agent znajduje miejsce dla sprzętów, które muszą stać pod ścianą, potem dla tych, które również muszą stać pod ścianą, ale mogą znajdować się pod oknem. Najpierw lodówka zostaje umieszczona przy którejś ze ścian (*location*(*fridge*) = 1) możliwie daleko od kuchenki. Szafka i obszar, gdzie będą znajdowały się kolejne szafki, zostają umieszczone obok siebie pod jedną ścianą lub w narożniku

dwóch ścian (*location*(*sideboard*) = 2, *location*(*storage*) = 2). Atrybutom hiperkrawędzi obiektowych i wierzchołków prawej strony produkcji zostają nadane wartości będące współrzędnymi specyfikującymi położenie sprzętów wewnątrz kuchni. Na końcu zostaje zaktualizowana wartość atrybutu określającego dostępną powierzchnię kuchni (atrybut hierarchicznej hiperkrawędzi prawej strony produkcji etykietowanej *Kitchen*).

Proces wartościowania hipergrafów podczas stosowania innych produkcji odbywa się w analogiczny sposób. ◊



Rys. 7.5. Pierwsza produkcja hierarchicznej gramatyki hipergrafowej agenta aranżującego kuchnię
Fig. 7.5. The first production of the hierarchical hypergraph grammar of the agent generating kitchen arrangements

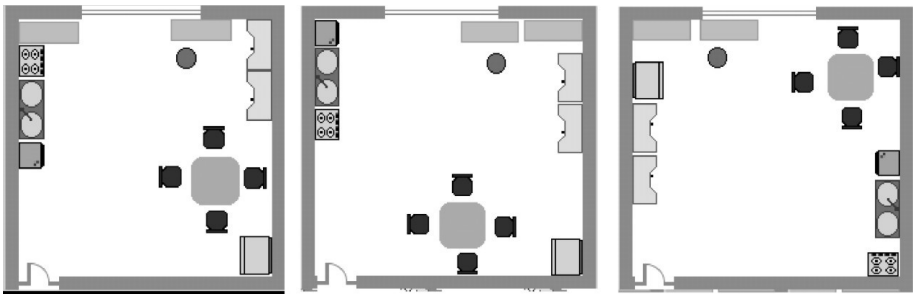
Hipergrafy otrzymywane w procesie wywodu i reprezentujące rozwiązania częściowe mogą być odwzorowane na modele graficzne z wykorzystaniem danej realizacji [GrŚP03]. Każdy agent projektowy odpowiedzialny za aranżacje pomieszczenia posiada w bazie wiedzy realizację, która umożliwia mu odwzorowanie wygenerowanego hipergrafu na model graficzny z wykorzystaniem odpowiednich ikon, a także predykaty, które specyfikują możliwe rozmiary i orientację prymitywów geometrycznych z bazy danych.

Ocena rozwiązań częściowych umożliwia agentowi wybór reguły gramatyki, która powinna być zastosowana w kolejnym kroku wywodu, podejmowanie decyzji dotyczących wymaganych modyfikacji rozwiązań oraz odnajdywanie nowych wymagań i koncepcji projektowych. Jeśli są potrzebne modyfikacje rozwiązania, agent może dodać do gramatyki reguły modyfikujące hipergrafy [Ślus04]. Jeśli pojawią się nowe idee lub wymagania projektowe, agent może utworzyć nowe reguły i dodać je do diagramu sterującego gramatyki [GGKŚ06]. Sprawdzanie rozwiązań częściowych w kolejnych krokach procesu projektowego pozwala agentowi poddać ocenie wiele wariantów rozmieszczenia elementów projektu i wybrać najbardziej odpowiedni kierunek poszukiwań rozwiązania.

Każdy agent projektujący aranżację pomieszczenia generuje zbiór rozwiązań spełniających początkowe wymagania. Zbiór ten może być traktowany jako zawartość długoterminowej pamięci agenta. W przypadku wywołania tego agenta przez agenta projektującego wnętrze domu kolejny raz z tymi samymi wymaganiami początkowymi może się on zachować w sposób reaktywny, zwracając jedno z wcześniej wygenerowanych rozwiązań. Jednak w przypadku nowych wymagań początkowych proces generacji rozwiązania musi zostać rozpoczęty od nowa.

Przykład 7.6: aranżacja pomieszczeń

Trzy różne rozkłady sprzętów w kuchni wygenerowane zgodnie z tymi samymi kryteriami (kącik jadalny ze stołem i krzesłami, zmywarka blisko zlewu, lodówka daleko od kuchenki, przynajmniej dwie szafki i jedna lada), czyli odpowiadające hierarchicznym hipergrafom o takiej samej strukturze, lecz różnych wartościach atrybutów specyfikujących lokalizację sprzętów, są przedstawione na rys. 7.6. ◊



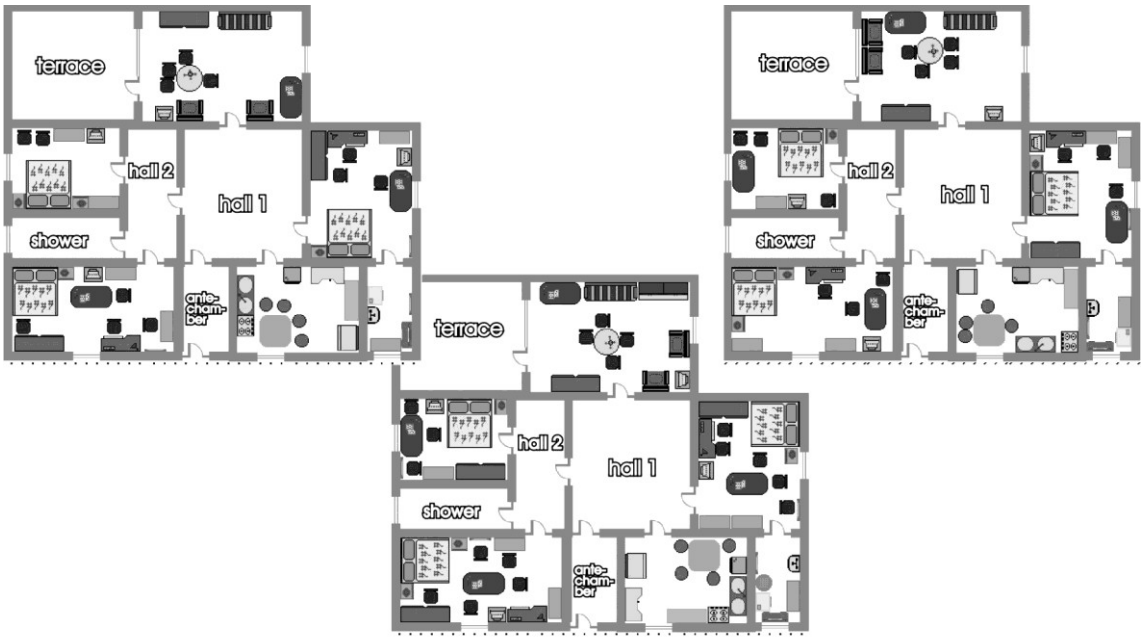
Rys. 7.6. Trzy różne rozkłady sprzętów kuchennych zgodne z tymi samymi kryteriami otrzymane w programie *Arrange Editor*

Fig. 7.6. Three different kitchen arrangements compatible with the same criteria obtained in *Arrange Editor* program

Działanie systemu agentów projektowych jest często czasochłonne ze względu na znaczną liczbę rozpatrywanych dopuszczalnych rozwiązań zadania projektowego. W niektórych przypadkach systemowi nie udaje się znaleźć poprawnego rozwiązania, nawet jeśli takie rozwiązanie istnieje, jednak kilkakrotne jego wywołanie prowadzi do znalezienia i przedstawienia projektantowi wielu interesujących, alternatywnych rozwiązań. Wyniki działania systemu agentów projektowych stanowią inspirację dla projektanta i wskazują na możliwość efektywnego wspierania człowieka na etapie projektowania koncepcyjnego. Agenci projektowi wykorzystujący bazę wiedzy i gramatyki hipergrafowe współpracując ze sobą dynamicznie reagują na zmiany zachodzące w kontekście projektowym i dopasowują generowane rozwiązania do nowych wymagań.

Przykład 7.7: aranżacja pomieszczeń

Trzy przykłady rozwiązań problemu zaprojektowania umeblowanego wnętrza domu otrzymane z wykorzystaniem pięciu współpracujących ze sobą agentów projektowych są pokazane na rys. 7.7. Trzy pokoje, salon, kuchnia i łazienka z rozkładu pomieszczeń domu z rys. 5.4a zostały wypełnione różnymi konfiguracjami mebli i sprzętów. ◊



Rys. 7.7. Trzy różne aranżacje pokoi dla tego samego rozkładu pomieszczeń otrzymane w programie *Arrange Editor*
Fig. 7.7. Three different furniture arrangements for the same floor layout obtained in *Arrange Editor* program

7.6. Podsumowanie

W rozdziale tym zostali omówieni agenci projektowi wykorzystujący wiedzę projektową do wspomagania projektanta na koncepcyjnym etapie projektowania. Ich działanie jest modelowane za pomocą hierarchicznych hipergrafowych gramatyk rozmieszczenia, a do oceny zgodności rozwiązań z kryteriami projektowymi jest stosowany logiczny mechanizm oparty na modalnym rachunku predykatów.

Została zaprezentowana architektura systemu składającego się ze zbioru komunikujących się ze sobą agentów, którzy autonomicznie rozwiązują powierzone sobie podzadania. Zostały przedstawione zasady funkcjonowania takiego systemu i sposób reprezentacji wiedzy agentów. Zbiór możliwych stanów środowiska agentów został zdefiniowany za pomocą języka hipergrafowego, który jest wspólnie generowany przez agentów projektowych. Zaprezentowany semantyczny model wieloagentowego systemu projektowego umożliwia agentom wykorzystanie modalnego rachunku predykatów do oceny stanów środowiska względem wyspecyfikowanych wymagań. Zostały zdefiniowane pojęcia poprawnego rozwiązania projektowego i rozwiązania zgodnego z określonymi kryteriami projektowymi.

Działanie przedstawionego systemu agentowego wspomagającego projektowanie zostało zilustrowane na przykładzie projektowania umeblowanej przestrzeni parterowego domu jednorodzinnego.

8. SYSTEMY WSPOMAGAJĄCE PROJEKTOWANIE

W rozdziale tym zostaną przedstawione systemy wspomagające koncepcyjny etap projektowania, które ułatwiają projektantowi podejmowanie decyzji dzięki wykorzystaniu wiedzy projektowej. Zostały one zaimplementowane w Zakładzie Projektowania i Grafiki Komputerowej UJ zgodnie z założeniami modelu teoretycznego przedstawionego w poprzednich rozdziałach pracy. Systemy te umożliwiają szybką generację i modyfikację wstępnych rozwiązań projektowych oraz wnioskowanie na temat ich poprawności.

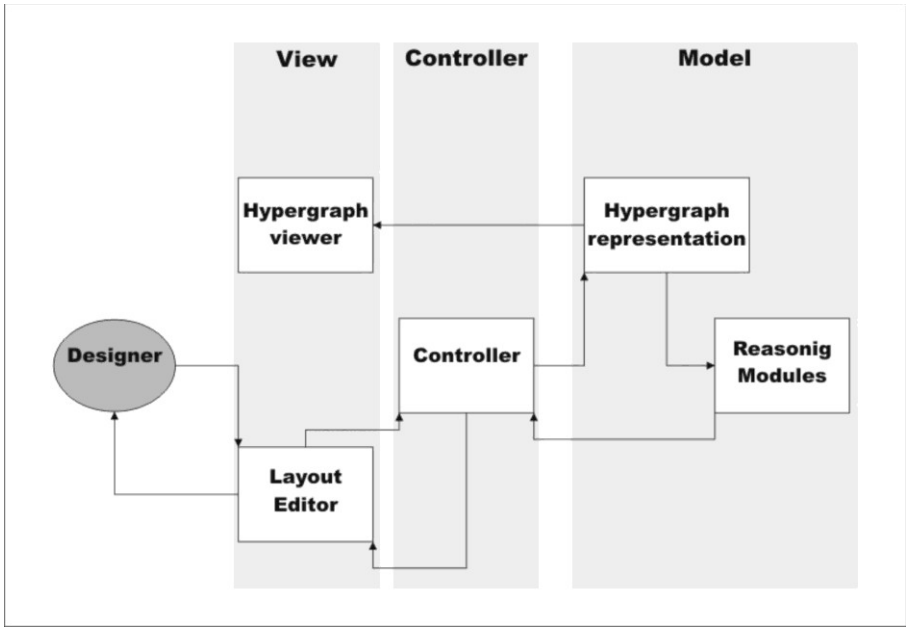
8.1. System projektowy wspomagający wnioskowanie

Prototypowy system wspomagający projektowanie i wnioskowanie *HSSDR* (Hypergraph System Supporting Design and Reasoning) został zaimplementowany przez Szymona Gajka w Javie [Gaje11]. System ten wykorzystuje wiedzę projektową i wspomaga użytkownika na etapie projektowania koncepcyjnego, pozwalając mu projektować rozkłady pomieszczeń z użyciem diagramów. Automatycznie jest generowana reprezentacja wewnętrzna diagramów w postaci hierarchicznych hipergrafów rozmieszczenia. System stanowi narzędzie pośredniczące pomiędzy językiem diagramowym a jego wewnętrzną reprezentacją. Podczas gdy użytkownik dokonuje edycji diagramów, system automatycznie wykonuje odpowiednie operacje na hipergrafach. Daje on także możliwość definiowania ograniczeń projektowych i wnioskowania dotyczącego poprawności diagramów względem tych ograniczeń poprzez sprawdzanie, czy są one spełnione przez hipergrafową reprezentację rozwiązań.

Architektura systemu *HSSDR* jest przedstawiona na rys. 8.1. System ten zawiera następujące moduły:

1. Graficzny interfejs użytkownika do edycji diagramów projektowych i ograniczeń związanych z ich komponentami (*Layout Editor*).
2. Generator wewnętrznej reprezentacji diagramów w postaci hierarchicznych hipergrafów (*Hypergraph representation*).
3. Moduł wizualizacji hipergrafów (*Hypergraph viewer*).

4. Moduł pozwalający wnioskować na temat diagramów poprzez sprawdzanie spełnialności wymagań i ograniczeń projektowych (*Reasoning Modules*).
5. Moduł zapewniający komunikację i synchronizację pomiędzy interfejsem graficznym i reprezentacją wewnętrzną (*Controller*).

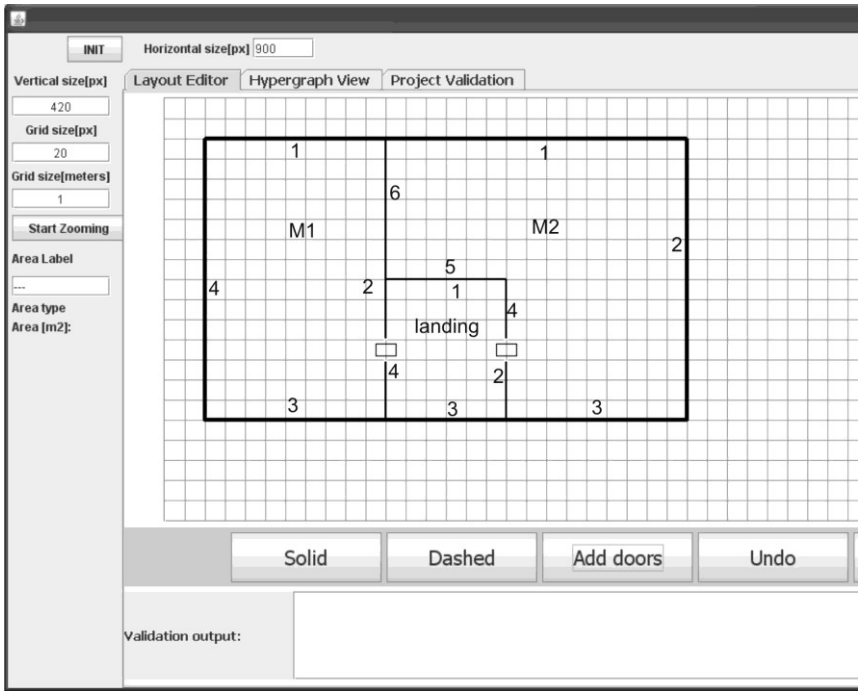


Rys. 8.1. Ogólna struktura systemu *HSSDR*
Fig. 8.1. The general structure of the *HSSDR* system

Interfejs graficzny pozwala użytkownikowi konstruować diagramy reprezentujące rozkłady pomieszczeń. Diagramy są rysowane na siatce, której rozmiar i rzeczywista odległość, jakiej odpowiada bok kwadratu siatki w projekcie, są ustalane przez użytkownika. Pierwszym krokiem jest narysowanie łamanej zamkniętej reprezentującej obrys projektowanego rozkładu pomieszczeń. Następnie powstały wielokąt jest dzielony na mniejsze wielokąty reprezentujące poszczególne pomieszczenia. Podział ten odbywa się poprzez narysowanie dowolnej łamanej pomiędzy dwoma punktami obrysu. Użytkownik może wprowadzać dowolne nowe pomieszczenia, zaczynając rysowanie łamanej na obrysie lub na jednej z nowo powstałych ścian. Na wszystkie odcinki reprezentujące ściany można w dowolnym momencie przeciągnąć pusty prostokąt symbolizujący drzwi. Powstające w ten sposób wielokąty reprezentują obiekty projektowe, a odcinki będące bokami tych wielokątów odpowiadają obiektom aktywnym zdefiniowanym w konceptualizacji projektowej. Widok okna edytora, w którym są rysowane diagramy reprezentujące rozkłady pomieszczeń, jest przedstawiony na rys. 8.2. Na projektowanym poddaszu został wydzielony podest schodów (*landing*) oraz dwa mieszkania (*M*1 i *M*2).

Edytor pozwala przypisywać komponentom diagramu etykiety i zbiory atrybutów, takich jak położenie czy powierzchnia pomieszczeń, długość i orientacja ścian. Interfejs graficzny pozwala użytkownikowi interakcyjnie tworzyć i edytować obiekty, ich atrybuty oraz wpro-

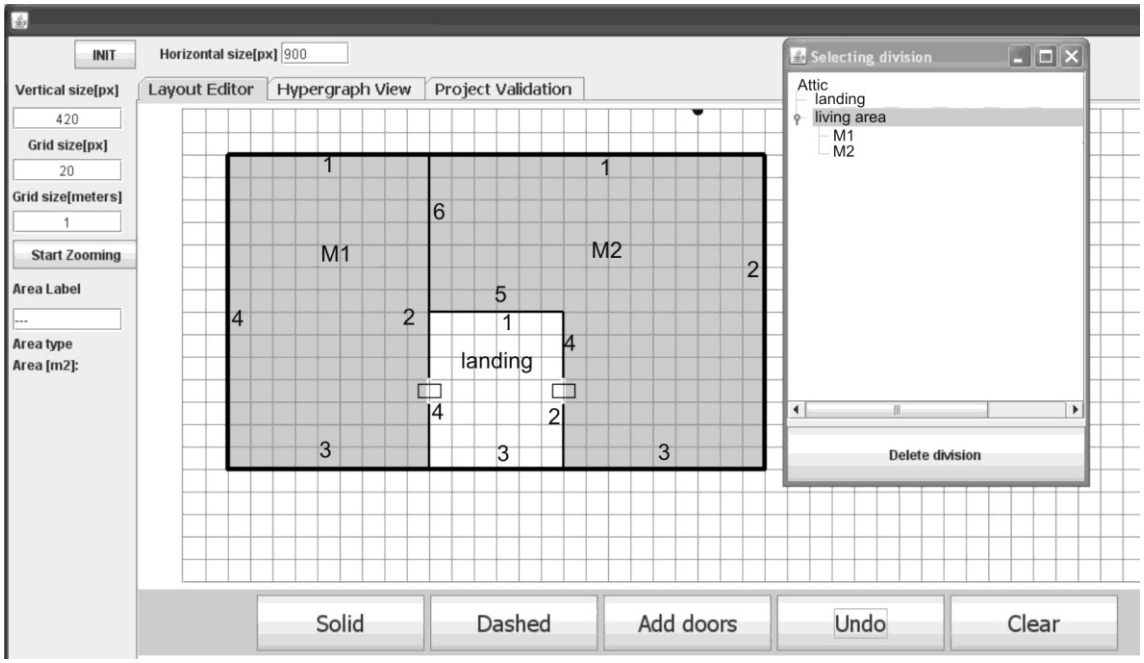
wadzać formuły wyrażające ograniczenia narzucone na strukturę diagramów i atrybuty obiektów reprezentowanych przez komponenty diagramów. Test sprawdzający dostępność pomiędzy mieszkaniami $M1$ i $M2$ jest analogiczny jak test sprawdzający dostępność między sypialnią i łazienką pokazany na rys. 6.5.



Rys. 8.2. .Przykładowy rozkład pomieszczeń na poddaszu
Fig. 8.2. A diagram of an attic floor arrangement

Edycja obiektów może być dokonywana przez bezpośrednią manipulację komponentami diagramu. Aby cofnąć wprowadzony wcześniej podział wielokąta, należy wybrać odpowiedni węzeł w drzewie hierarchii reprezentującym dokonane podziały. Wówczas na diagramie zostaje podświetlony wielokąt odpowiadający podziałowi reprezentowanemu przez ten węzeł, czyli wielokąt, który przed podziałem reprezentował jedno pomieszczenie lub obszar. Podział tego wielokąta zostanie usunięty. Wybór węzła drzewa i odpowiadający mu wielokąt, którego podział zostanie usunięty, są przedstawione na rys. 8.3. Wartości atrybutów, takich jak powierzchnia pomieszczeń, długość i porządek ścian są ustalane automatycznie przez system i wynikają bezpośrednio z rozmiaru kwadratu siatki ustalonego przez użytkownika i kształtu narysowanych wielokątów.

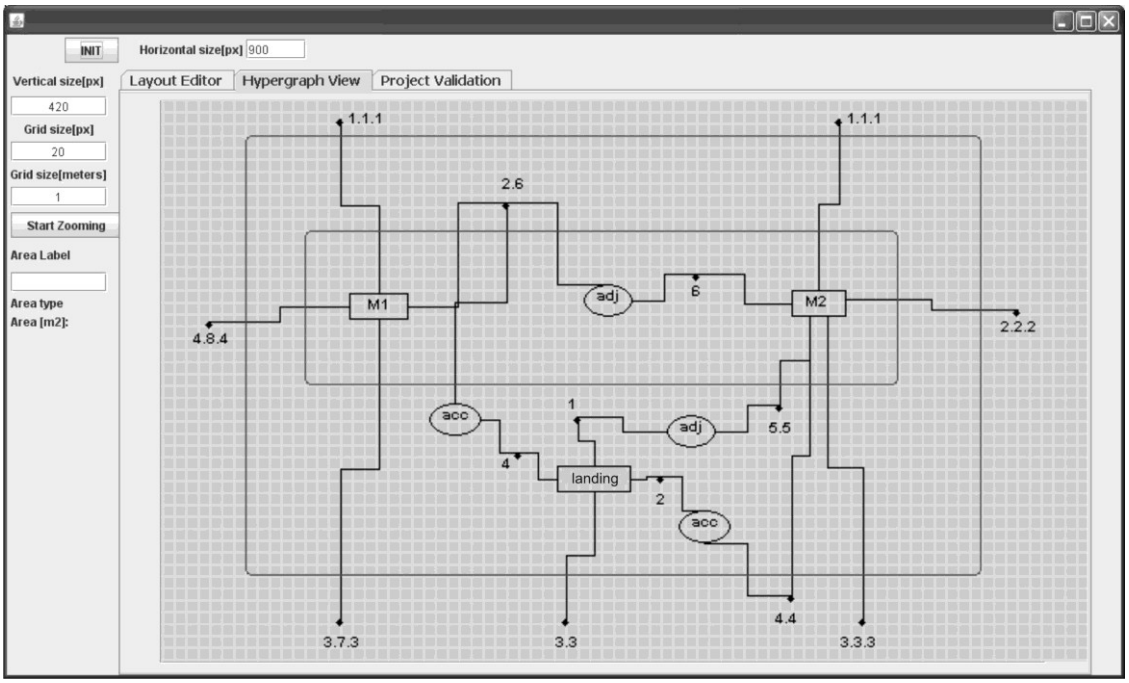
Generator wewnętrznej reprezentacji diagramów automatycznie tworzy atrybutowane hierarchiczne hipergrafy rozmieszczenia, których hiperkrawędzie odpowiadają komponentom diagramów oraz relacjom pomiędzy tymi komponentami. Moduł ten wykonuje na hipergrafach operacje odpowiadające akcjom projektowym wykonywanym przez użytkownika na diagramach.



Rys. 8.3. Wybór wielokąta, którego podział zostanie usunięty
Fig. 8.3. Choosing a polygon which division will be removed

Moduł wizualizacji hierarchicznych hipergrafów rozmieszczenia weryfikuje i aktualizuje hierarchię oraz umożliwia grupowanie komponentów należących do jednego poziomu hierarchii i przedstawianie relacji między tymi grupami i komponentami znajdującymi się na innym poziomie hierarchii hipergrafów. Istniejące drzewo hierarchii umożliwia nawigację pomiędzy różnymi poziomami hierarchii hipergrafu. Dzięki temu projektant może wybrać hiperkrawędzie, które powinny być wizualizowane i zdecydować, czy ich zawartość ma być także widoczna. Hierarchiczny hipergraf reprezentujący rozkład pomieszczeń z rys. 8.2 jest pokazany na rys. 8.4.

Moduł wnioskowania zawiera zarówno formuły logiczne pierwszego rzędu charakteryzujące diagramy projektowe, jak i formuły wyrażające wymagania i ograniczenia, jakie powinny spełniać diagramy projektowe. Formuły te są zbudowane nad zbiorem symboli stałych reprezentujących obiekty projektowe, symboli funkcji reprezentujących atrybuty obiektów i operatory działające na tych atrybutach oraz symboli predykatów reprezentujących relacje między obiektami i warunki wiążące atrybuty obiektów. Moduł wnioskowania operuje na reprezentacji hipergrafowej umożliwiającej bezpośredni dostęp do informacji o komponentach diagramów, ich atrybutach i relacjach między komponentami. Na podstawie tych informacji tworzy on strukturę relacyjną nadającą znaczenie formułom logicznym opisującym diagramy i pozwalającą sprawdzić, czy formuły logiczne wyrażające kryteria projektowe są spełnione przez diagramy projektowe. Moduł ten steruje transformacją diagramów podpowiadając użytkownikowi, jakie modyfikacje są niezbędne.



Rys. 8.4. Hierarchiczny hipergraf reprezentujący rozkład pomieszczeń z rys. 8.2
Fig. 8.4. A hierarchical hypergraph representing the floor layout from fig. 8.2

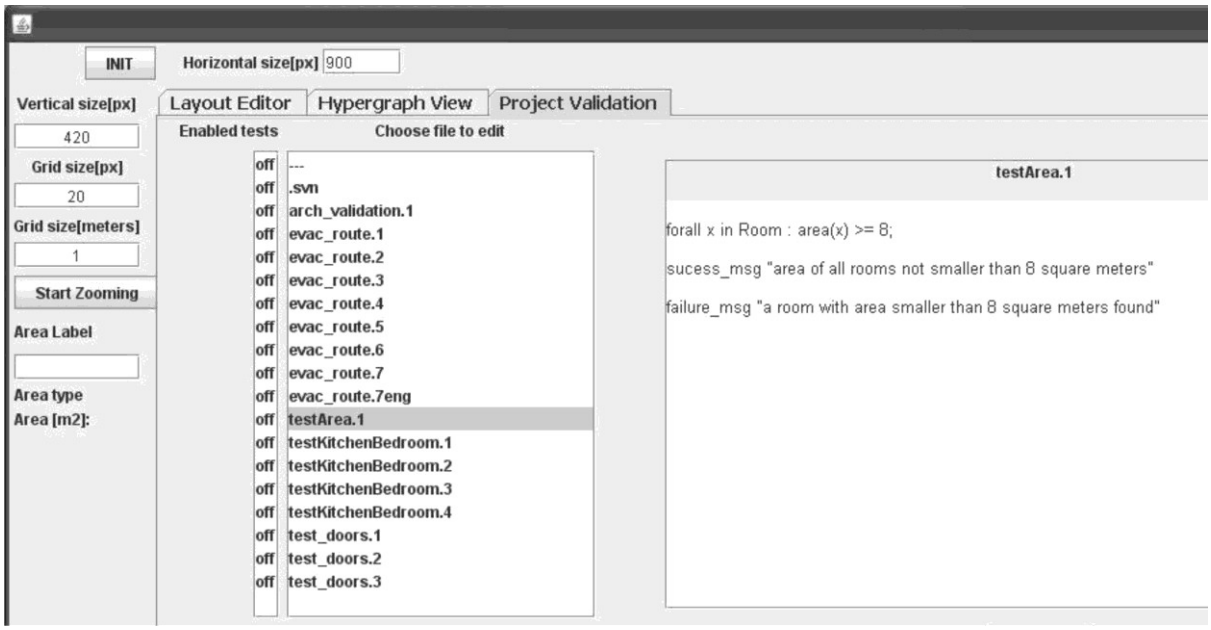
Moduł wnioskowania na bieżąco monitoruje fizyczne akcje projektowe wykonywane przez użytkownika, aktualizuje wiedzę o diagramie zawartą w formułach logicznych, analizuje aktualny stan projektowanego diagramu, porównując formuły wyrażające wymagania projektowe z formułami opisującymi diagram i powiadamia użytkownika o ewentualnym naruszeniu ograniczeń projektowych. Po każdej modyfikacji rozkładu pomieszczeń powodującej zmianę struktury lub atrybutów hipergrafu są uruchamiane testy sprawdzające poprawność projektu. Jeśli kończą się one niepowodzeniem, użytkownikowi jest przekazywana odpowiednia informacja i są sugerowane możliwe sposoby wyeliminowania błędów. Rysunek 8.5 przedstawia jeden z przykładowych testów systemu wykorzystujących wartości atrybutów przypisanych atomom hipergrafu reprezentującego diagram. Pozwala on sprawdzić, czy powierzchnia wszystkich pokoi jest co najmniej równa 8 m^2.

8.2. Wieloagentowy system projektowy

Prototypowy wieloagentowy system projektowy *Arrange Editor* został zaimplementowany przez Tomasza Zarasia w języku Java, a interfejs użytkownika jest oparty na bibliotece Standard Widget Toolkit (SWT) [Zara09].

Jest to modularny system współbieżny pomagający projektantowi w rozwiązywaniu złożonych zadań projektowych. System składa się ze zbioru komunikujących się ze sobą agentów, którzy autonomicznie rozwiązują powierzone sobie podzadania, operując na hierarchicznej hipergrafowej reprezentacji projektów. Każdy agent posiada zarówno odpowiednią

hierarchiczną hipergrafową gramatykę rozmieszczenia, jak i bazę wiedzy zawierającą fakty dotyczące podzadania, za które jest on odpowiedzialny.



Rys. 8.5. Test sprawdzający powierzchnię pomieszczeń
Fig. 8.5. A test checking the area of rooms

Dzięki wykorzystywanej reprezentacji projektowanych obiektów w postaci hierarchicznych hipergrafów rozmieszczenia agenci mogą się w łatwy sposób adaptować do zmian specyfikacji problemu projektowego zachodzących w trakcie projektowania. Wiedza projektowa zakodowana w atrybutowanych hierarchicznych hipergrafach pozwala im dokonywać oceny rozwiązań częściowych reprezentujących aktualny stan projektowanego obiektu i podejmować na tej podstawie kolejne decyzje projektowe. Agenci planują swoje przyszłe zachowanie poprzez wybór produkcji gramatyki, które należy zastosować w kolejnych krokach procesu projektowego. Na wybór ten mają wpływ związane z produkcjami predykaty ich stosowalności, które odzwierciedlają kryteria projektowe.

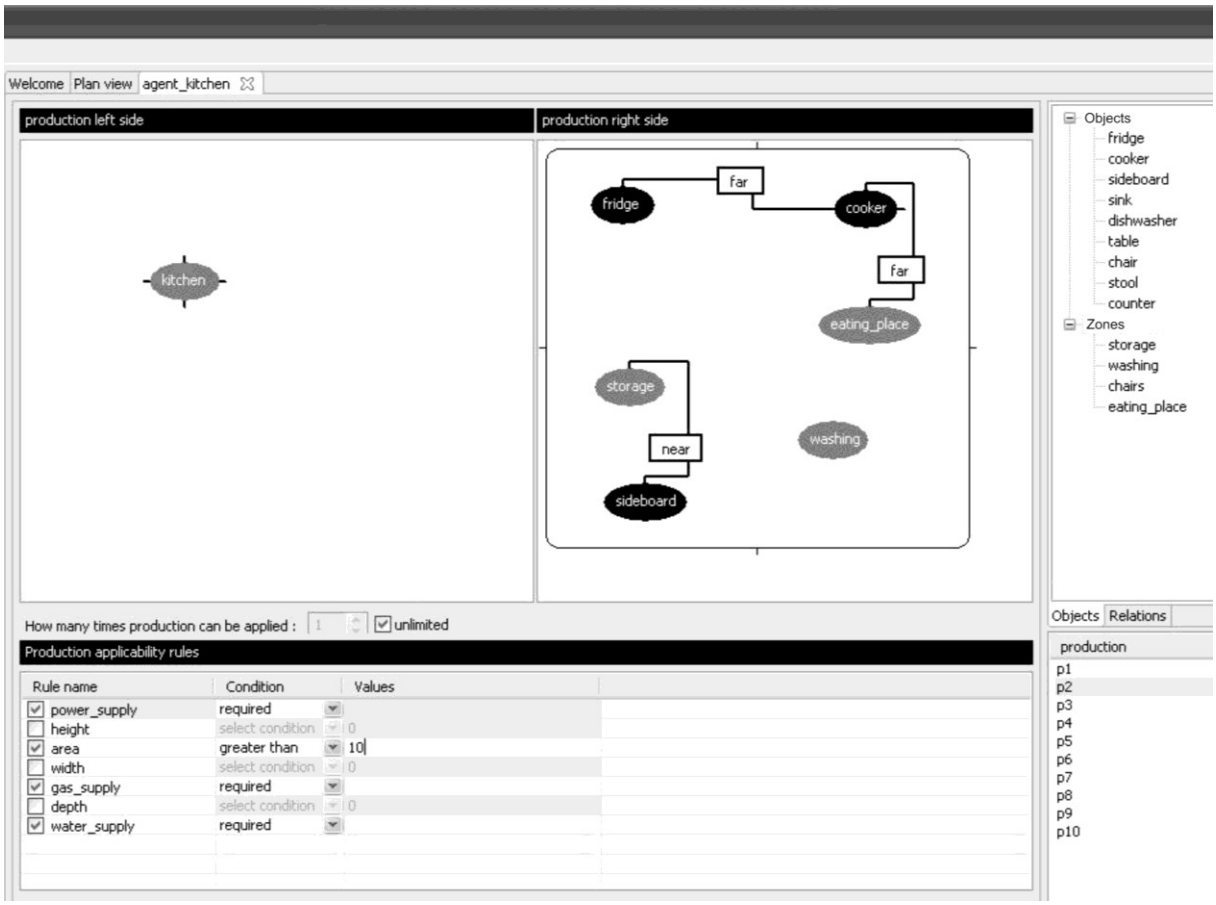
Aplikacja składa się z dwóch modułów:

1. Edytora służącego do tworzenia oraz modyfikowania hierarchicznych hipergrafowych gramatyk rozmieszczenia wraz z diagramami sterowania i predykatami stosowalności określającymi wartości odpowiednich atrybutów atomów hipergrafów.
2. Generatora służącego do tworzenia wizualizacji rozwiązań odpowiadających hipergrafom wygenerowanym przez cały system lub poszczególnych agentów.

Moduł edytora pozwala wybrać rodzaj agenta, a następnie zdefiniować hierarchiczną hipergrafową gramatykę rozmieszczenia modelującą jego zachowanie. Wraz z wyborem agenta w głównej części okna aplikacji pojawia się zakładka dla tego agenta. Użytkownik aplikacji może tworzyć nowe produkcje, modyfikować lub usuwać już istniejące. Wraz z tworzeniem produkcji definiuje się predykaty ich stosowalności. Określane są wartości atrybutów przypi-

sanych hiperkrawędziom i warunki, jakie powinny one spełniać. Warunki wiążące atrybuty stanowią predykat stosowalności. Po utworzeniu produkcji w tej samej zakładce jest tworzony diagram sterujący dla gramatyki.

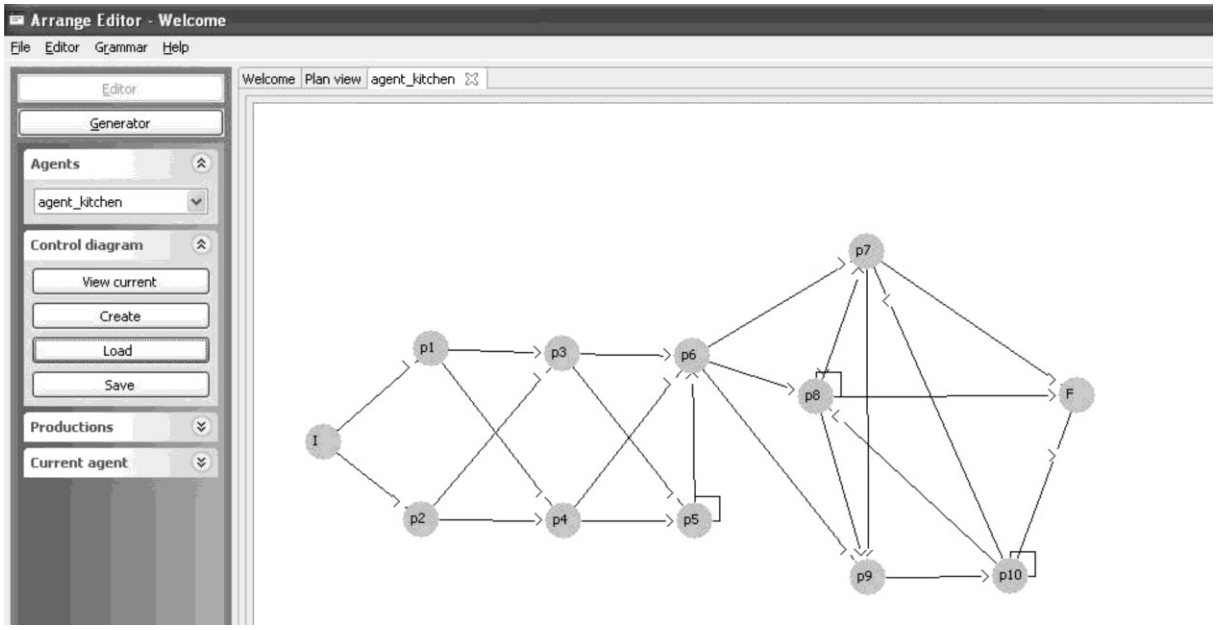
Zakładka, w której można stworzyć hierarchiczną hipergrafową gramatykę rozmieszczenia dla agenta odpowiedzialnego za aranżację kuchni, jest przedstawiona na rys. 8.6. Jedna z produkcji jest pokazana w jej górnym środkowym obszarze, który służy do definiowania poszczególnych produkcji. Górny obszar po prawej stronie zawiera dostępne obiekty reprezentujące meble oraz obszary lub grupy mebli. Obiekty te będą reprezentowane w produkcjach gramatyki hipergrafowej odpowiednio jako hiperkrawędzie bez zawartości i hiperkrawędzie hierarchiczne. W tym samym oknie mogą zostać wyświetlone relacje, jakie mogą zachodzić pomiędzy dostępnymi obiektami. Relacje te są reprezentowane w hipergrafach za pomocą hiperkrawędzi relacyjnych. Dolny obszar po prawej stronie zawiera listę dotychczas utworzonych produkcji dla danego agenta. Dolny środkowy obszar zakładki pozwala nadać wartości atrybutom przypisanym obiektom i dodać do bazy wiedzy agenta ograniczenia dotyczące tych atrybutów. Jeśli ograniczenia te dotyczą atrybutów lewej strony produkcji, wówczas stanowią predykat stosowalności.



Rys. 8.6. Zakładka, w której jest definiowana gramatyka dla agenta aranżującego kuchnię
Fig. 8.6. A window in which a grammar of the kitchen arrangement agent is defined

Na rys. 8.6 są widoczne atrybuty: *power_supply*, *gas_supply*, *water_supply*, *height*, *area*, *width*, *depth*. Pierwsze trzy z nich przyjmują wartości logiczne. W tym przypadku dostęp do energii elektrycznej, instalacji gazowej i wodnej jest wymagany i należy określić, jakich sprzętów reprezentowanych przez hiperkrawędzie prawej strony produkcji ten warunek dotyczy. Pozostałe atrybuty określają rozmiary i powierzchnię obiektów reprezentowanych przez hiperkrawędzie, i przyjmują wartości liczbowe, do których można zastosować operatory arytmetyczne i relacyjne.

Zawartość zakładki zmienia się podczas definiowania i edycji diagramu sterującego. Diagram ten jest tworzony w środkowym oknie zakładki (rys. 8.7) z produkcji wyszczególnionych w dolnym oknie po prawej stronie (rys. 8.6).



Rys. 8.7. Okno, w którym jest definiowany diagram sterujący dla hierarchicznej gramatyki hipergrafowej agenta aranżującego kuchnię

Fig. 8.7. The window in which a control diagram for the hierarchical hypergraph grammar of the kitchen arrangement agent is defined

Wyniki działania generatora rozwiązań są wyświetlane w zakładce *Plan view* (rys. 8.8) przypisanej agentowi-menadżerowi. Zakładka ta może zawierać rozwiązanie (wizualizację wygenerowanego hipergrafu) będące wynikiem działania wszystkich agentów lub tylko wybranych agentów. Rysunek 8.8 przedstawia wizualizację hipergrafu rozmieszczenia wygenerowanego przez agenta projektującego rozkład pomieszczeń wraz ze współpracującymi z nim agentami aranżującymi pomieszczenia.

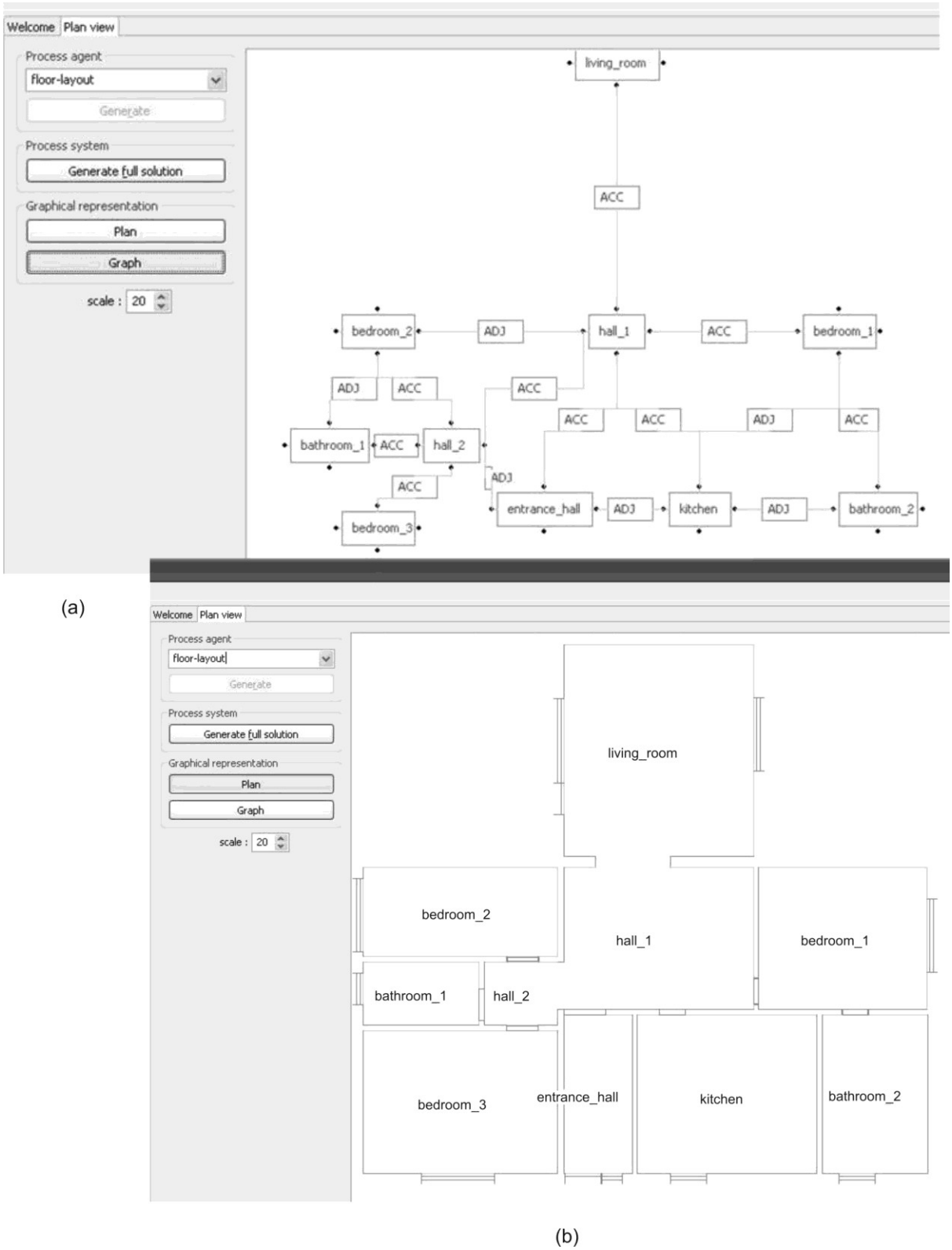
Wyniki działania poszczególnych agentów w postaci wygenerowanych przez nich hierarchicznych hipergrafów pojawiają się w nowych zakładkach etykietowanych nazwami agentów. Podczas stosowania produkcji agenci nadają wartości atrybutom określającym położenie, które są przypisane hiperkrawędziom obiektowym reprezentującym sprzęty i meble występujące w pomieszczeniach, w taki sposób aby zachowane były odpowiednie relacje mię-

dzy meblami reprezentowane przez hiperkrawędzie relacyjne, były spełnione wymagania dotyczące ich lokalizacji oraz aby meble te nie zachodziły na siebie. Znając wymiary mebli znajdujących się w bazie wiedzy, agenci wykorzystują do obliczania odpowiednich wartości atrybutów algorytm plecakowy [CLRD03]. Graficzna reprezentacja wygenerowanych hipergrafów, w której hiperkrawędziom obiektowym odpowiadają meble, pojawia się w zakładce *Plan view*.

Rys. 8.8. Rozwiązanie wygenerowane przez agenta projektującego rozkład pomieszczeń i agentów aranżujących pomieszczenia
Fig. 8.8. A solution generated by the agent designing floor layout and agents arranging rooms

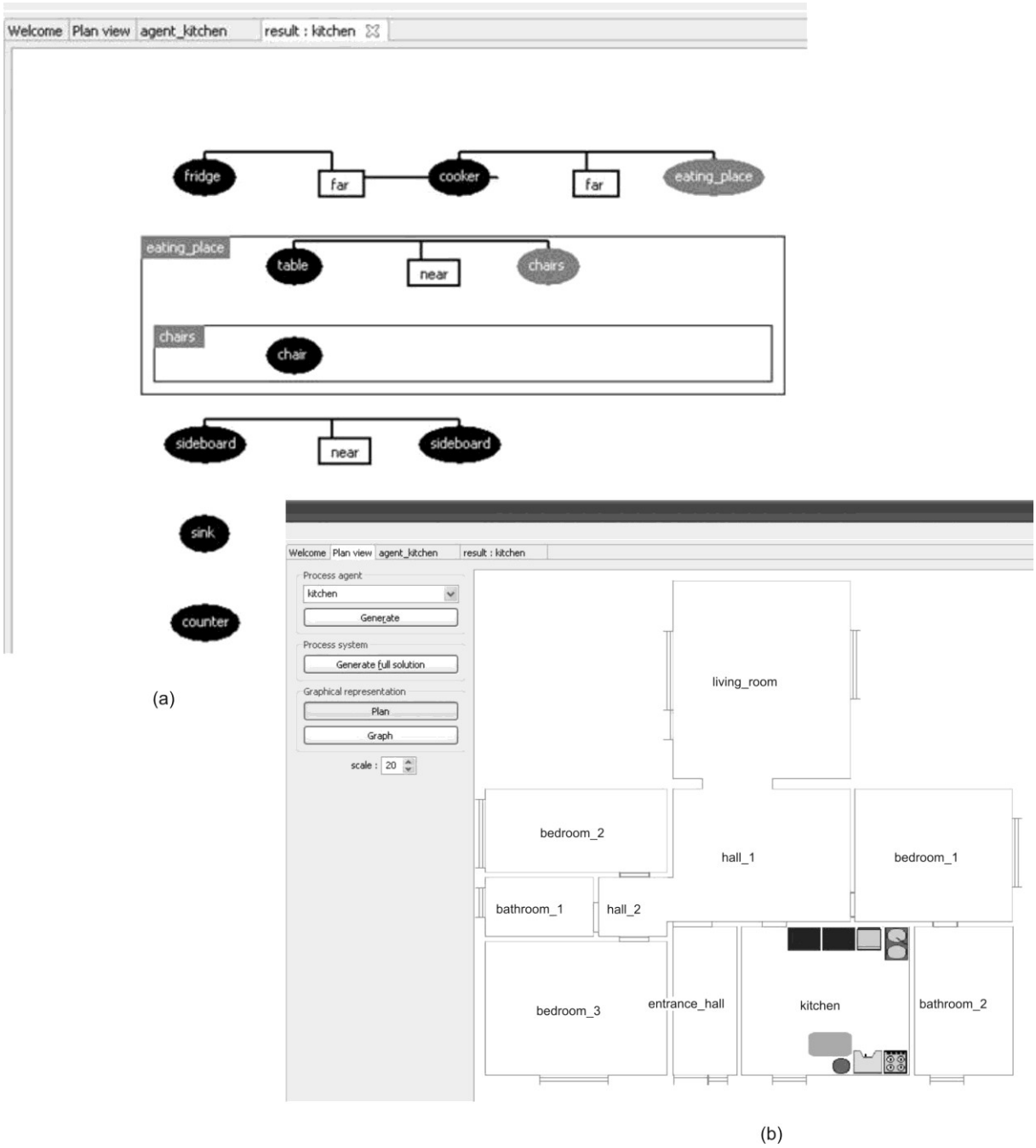
Plan mieszkania, będący wynikiem działania agenta projektującego rozkład pomieszczeń, jest przedstawiony na rys. 8.9b, a hipergraf, na podstawie którego został narysowany ten plan, na rys. 8.9a.

Hierarchiczny hipergraf wygenerowany przez agenta aranżującego kuchnię oraz rozkład pomieszczeń z kuchnią umeblowaną i wyposażoną zgodnie z tym hipergrafem są przedstawione odpowiednio na rys. 8.10a i 8.10b. Rozkład pomieszczeń z rys. 8.10b odpowiada hipergrafowi z rys. 8.9a, w którym hiperkrawędź etykietowana *kitchen* została zastąpiona hipergrafem z rys. 8.10a.



Rys. 8.9. Rozkład pomieszczeń: a) hipergraf reprezentujący rozkład, b) wynik działania agenta projektującego rozkład pomieszczeń

Fig. 8.9. A floor layout: a) a hypergraph representing the layout, b) the result of the floor layout arrangement agent action



Rys. 8.10. Aranżacja pomieszczeń: a) hipergraf reprezentujący rozkład sprzętów w kuchni, b) rozkład pomieszczeń z kuchnią umeblowaną zgodnie z tym hipergrafem

Fig. 8.10. Room arangement: a) a hypergraph representing the arrangement of kitchen pieces, b) the floor layout with the kitchen arranged according to this hypergraph

9. PODSUMOWANIE

W czasach gdy komunikacja wizualna stała się integralną częścią naszej codziennej rzeczywistości, rośnie znaczenie diagramów i rysunków w procesie kreatywnego myślenia, podejmowania decyzji i rozwiązywania problemów. Komputerowe systemy projektowania wykorzystujące języki wizualne, które pozwalają na bieżąco śledzić zmiany dokonywane w projektach, powinny być wyposażone w inteligentne narzędzia wspomagające użytkownika podczas procesu projektowego. Narzędzia takie wykorzystują wiedzę projektową i bazują na mechanizmach wnioskowania na podstawie wewnętrznej reprezentacji elementów języka wizualnego.

Przedstawiona rozprawa prezentuje formalny model interakcyjnego systemu wspomagającego innowacyjne projektowanie wizualne, w którym dialog między projektantem a systemem odbywa się za pomocą języka wizualnego. System ten jest wyposażony w agentów projektowych generujących i sugerujących projektantowi potencjalne, poprawne rozwiązania projektowe. Została zaproponowana konceptualizacja będąca formalną reprezentacją wiedzy z dziedziny projektowania wizualnego i dostarczająca kategoryzacji i hierarchizacji pojęć z tej dziedziny. Zapewnia ona właściwą interpretację wykorzystywanej wiedzy i daje możliwość jej modyfikacji bez utraty spójności. Bazujący na tej konceptualizacji diagramowy język projektowania pozwala na jednoznaczne rozważanie problemów projektowych. Formalizacja wiedzy projektowej umożliwia też przekształcanie reprezentacji projektów, np. do modelu IFC będącego obecnie standardowym formatem danych dla większości narzędzi CAD-owskich, co zapewnia swobodną wymianę projektów między architektami, inżynierami i konstruktorami.

Została omówiona wprowadzona w pracy syntaktyka i semantyka diagramowego języka projektowania, który stanowi narzędzie pośredniczące pomiędzy wiedzą z danej dziedziny a jej wewnętrzną reprezentacją. Projektant ma możliwość korzystania z komputerowego systemu projektowego za pośrednictwem języka wizualnego, zarówno poprzez edycję diagramów, jak i specyfikację ograniczeń projektowych, bez potrzeby znajomości wewnętrznej reprezentacji tego języka. Reprezentacja wewnętrzna diagramów w formie hierarchicznych hipergrafów jest przekształcana zarówno z wykorzystaniem narzędzia generacyjnego w postaci

zdefiniowanych tu gramatyk hipergrafowych, jak i operacji na hipergrafach pozwalających na efektywne dokonywanie modyfikacji odpowiadających podejmowanym akcjom projektowym. Zastosowanie hierarchicznych hipergrafów odzwierciedla metodę projektowania „od ogółu do szczegółu", umożliwia rozważanie projektu na wybranym poziomie szczegółowości oraz pozwala na badanie relacji zachodzących pomiędzy fragmentami obiektu znajdującymi się na różnych poziomach hierarchii.

Wykorzystanie atrybutowanych hierarchicznych hipergrafów jako reprezentacji wewnętrznej struktur projektów odpowiadających diagramom pozwala także na efektywne przechowywanie wiedzy projektowej w systemie. Reprezentacja ta jest więc bogata semantycznie i łatwa do automatycznego przetwarzania. Taki rodzaj reprezentacji wewnętrznej diagramów stanowi dogodną podstawę do zdefiniowania mechanizmu automatycznego wnioskowania o projektach. Diagramy projektowe, będące elementami języka wizualnego, są automatycznie przekształcane na hierarchiczne hipergrafy, a następnie informacja zawarta w tych hipergrafach jest tłumaczona na formuły logiczne pierwszego rzędu. Formuły te są definiowane nad wielorodzajowym słownikiem, którego symbole są odwzorowywane w elementy konceptualizacji projektowej za pomocą dopasowania ontologicznego. Dzięki temu symbole słownika odnoszą się do obiektów projektowych reprezentowanych przez komponenty diagramów i do ich własności. Formuły logiczne w sposób formalny kodują wiedzę o projektach, będących zarówno kompletnymi, jak i częściowymi rozwiązaniami. Przedstawiona struktura relacyjna, która przypisuje składowym formuł logicznych elementy hipergrafów i wartości ich atrybutów, określa semantykę tych formuł i pozwala systemowi śledzić zmiany w projektach, zachodzące na skutek podejmowanych akcji projektowych oraz wyciągać istotne wnioski dotyczące projektów. Zbiór formuł logicznych opisujących projekt jest porównywany ze zbiorem formuł pierwszego rzędu wyrażających ograniczenia i wymagania projektowe zarówno stanowiące ogólną wiedzę projektową, jak i zdefiniowane przez projektanta.

Omówiony w rozprawie logiczny model wnioskowania dotyczącego diagramów projektowych pozwala systemowi wspomagającemu projektowanie sprawdzać kompatybilność rozwiązań z normami i standardami projektowymi oraz sugerować projektantowi korzystne modyfikacje tworzonych rozwiązań. Szczególnie semantyczna analiza diagramów dostarcza nowej jakości w językach wizualnych, ponieważ pozwala w bardziej szczegółowy sposób modelować ograniczenia oparte na wiedzy projektowej. Badanie poprawności projektu może również zachodzić na poziomie reprezentacji wewnętrznej, gdyż ograniczeniom i wymaganiom projektowym odpowiadają także określone warunki zdefiniowane dla hierarchicznych hipergrafów reprezentujących diagramy. Sprawdzanie, czy hipergrafy spełniają te warunki, odpowiada stosowaniu dobrze określonych zasad projektowania inżynierskiego lub architektonicznego.

Przedstawione w pracy podejście łączące wizualizację projektów z ich wewnętrzną reprezentacją stanowi także model formalny dla systemu inteligentnych asystentów wspomagających projektanta na etapie projektowania koncepcyjnego. Zaproponowany tu wieloagentowy system projektowy składa się ze zbioru komunikujących się ze sobą agentów, którzy autonomicznie rozwiązują powierzone sobie podzadania, operując na hierarchicznej hipergrafowej reprezentacji projektów. Zachowanie agentów jest modelowane za pomocą zbioru reguł hierarchicznych hipergrafowych gramatyk rozmieszczenia i bazy wiedzy zawierającej fakty dotyczące rozwiązywanego zadania.

Został omówiony semantyczny model wieloagentowego systemu projektowego wykorzystującego gramatyki hipergrafowe. Do oceny poprawności rozwiązań i ich zgodności z kryteriami projektowymi jest wykorzystywany wprowadzony tu mechanizm wnioskowania bazujący na modalnym rachunku predykatów zdefiniowanych nad wielorodzajowym słownikiem. Wiedza projektowa zakodowana w atrybutowanych hierarchicznych hipergrafach pozwala agentom adaptować się do zmian specyfikacji problemu projektowego zachodzących w trakcie projektowania. Możliwość oceny rozwiązań częściowych reprezentujących aktualny stan projektowanego obiektu umożliwia agentom podejmowanie decyzji projektowych i planowane przyszłego zachowania. Przedstawiony system wieloagentowy generuje język będący zbiorem hipergrafów reprezentujących różne rozwiązania projektowe kompatybilne z narzuconymi kryteriami projektowymi. Realizacja wygenerowanych hipergrafów pozwala projektantowi dokonywać oceny modeli graficznych stworzonych projektów.

Przedstawione w rozprawie narzędzia zastosowane w prototypowych systemach wspomagania projektowania opartych na wiedzy projektowej wskazują na możliwość efektywnego asystowania w procesie innowacyjnego projektowania poprzez ułatwienie projektantowi szybkiej generacji i modyfikacji wstępnych rozwiązań projektowych oraz wnioskowanie na temat ich poprawności. Zaproponowane metody formalizacji i kodowania wiedzy projektowej, automatycznego przetwarzania wewnętrznej reprezentacji projektowanych obiektów, specyfikacji ograniczeń i wymagań projektowych oraz wnioskowania dotyczącego tworzonych rozwiązań w znacznym stopniu pomagają projektantowi w podejmowaniu istotnych decyzji projektowych. Podczas gdy istniejące systemy sprawdzają poprawność kompletnych rozwiązań względem wyspecyfikowanych reguł projektowych, omawiane narzędzie pozwala na weryfikację projektów na kolejnych etapach rozwoju rozwiązania. Określona konceptualizacja zadaniowa i odpowiedni dobór reguł projektowych sprawią, że systemy wspomagające koncepcyjną fazę procesu projektowego znajdą zastosowanie w projektowaniu architektonicznym i aranżacji wnętrz, a także w projektowaniu przedmiotów użytkowych i urządzeń mechanicznych.

Kolejnym krokiem pracy badawczej będzie połączenie obu przedstawionych podejść do tworzenia systemów wspomagania projektowania, bazujących na wiedzy zawartej w hierar-

chicznych hipergrafach, a więc podejścia wykorzystującego agentów projektowych do generowania poprawnych rozwiązań wizualizowanych jako diagramy projektowe i metody wnioskowania o projektach na podstawie diagramów projektowych interaktywnie tworzonych przez użytkownika. Do systemu *HSSDR* wspomagającego projektowanie i wnioskowanie zostanie dodany moduł służący do tworzenia agentów i wyposażenia ich w hierarchiczne gramatyki grafowe działające na wewnętrznej reprezentacji diagramów wykorzystywanych w tym systemie. Dodatkowo moduł wnioskowania służący do oceny poprawności rozwiązań generowanych przez agentów, który bazuje na modalnych predykatach, zostanie rozszerzony dla pełnej logiki modalnej pierwszego rzędu, tzn. również dla formuł zawierających zmienne i kwantyfikatory.

Agenci, oprócz stosowanych dotychczas gramatyk grafowych, będą posiadali także gramatyki opisowe, co pozwoli im określić jakościową i ilościową informację o rozwiązaniach generowanych przez innych agentów. Działając na globalnej bazie danych zawierającej wcześniej stworzone rozwiązania, będą mogli wyszukać najbardziej zbliżone projekty, oszacować koszty wybudowania i wyposażenia zaprojektowanych budynków oraz urządzenia zaprojektowanych ogrodów, ocenić dostępność potrzebnych materiałów i roślin ogrodowych. Agenci zostaną także wyposażeni w funkcję uczącą. Na podstawie ocen przypisanych hipergrafom reprezentującym wygenerowane rozwiązania przez innych agentów, środowisko albo projektanta, agent dowie się, które sekwencje produkcji gramatyki są preferowane i zmodyfikuje diagram sterujący poprzez uaktualnianie wag określających prawdopodobieństwo zastosowania produkcji. W obecnej wersji systemu agenci współpracują i informują się wzajemnie o niemożliwości znalezienia prawidłowego rozwiązania, nie prowadzą jednak negocjacji. W przyszłości chcielibyśmy, aby agenci posiadali wpływ na decyzje innych agentów.

Zaproponowany model interakcyjnego projektowania wizualnego zostanie przetestowany na innych zadaniach projektowych z dziedziny architektury. W kolejnych aplikacjach zostaną stworzone zorientowane problemowo języki wizualne, służące projektowaniu trójwymiarowych form budynków i obiektów o określonych stylach architektonicznych. Zostanie także opracowany sposób efektywnego tworzenia bazy formuł logicznych reprezentujących ograniczenia i wymagania projektowe na podstawie obowiązujących norm i standardów projektowych. Baza ta zostanie następnie uporządkowana i zostanie określona kolejność, w jakiej będzie sprawdzana poprawność projektu względem bazy formuł.

BIBLIOGRAFIA

[AtGG10] AGG – *Attributed Graph* Grammars, 2010, http://tfs.cs.tu-berlin.de/agg/.

[AjTs08] Ajiro T., Tsuchida K.: *Visual Programming Language for Bit-Level Concurrent Programming: APECbits*, IEEE Symposium on Visual Languages and Human-Centric Computing, Germany, 2008.

[AkMo04] Akin O., Moustapha H.: *Formalizing Generation and Transformation in Design*, in Gero J. S. (Ed.), Design Computing and Cognition'04, Kluwer Academic Publishers, Dodrecht 2004, s. 177÷196.

[Allp10] *Allplan Architecture*, <http://www.allplan.com/>, Nemetschek, 2010.

[Arch69] Archer, B. L.: *The structure of the design process*, in Broadbent G., Ward A. (Eds.), Design Methods in Architecture, Lund Humphries, London 1969, s. 76÷102.

[Arch79] Archer, B. L.: *Whatever Became of Design Methodology*?, Design Studies 1, 1979, s. 17÷18.

[ArCA10] *ArchiCAD*, <http://www.graphisoft.com/products/archicad/>, Graphisoft, 2010.

[Auto02] *Autodesk Architectural Desktop 3.3 User's Guide*, Autodesk, 2002.

[AuRA10] *Autodesk Revit Architecture*, <http://www.autodesk.com/revit>, Autodesk, 2010.

[BaNi99] Baader F., Nipkow T.: *Term Rewriting and All That*, Cambridge 1999.

[BaCo05] Banyasad O., Cox P. T.: *Integrating Design Synthesis and Assembly of Structured Objects in a Visual Design Language*, Theory and Practice of Logic Programming 5, 2005, s. 601÷621.

[BaEh99] Bardohl R., Ehrig H.: *Conceptual Model for the Graphical Editor GenGEd for the Visual Definition of Visual Languages*, in Ehrig H., Engels G., Kreowski H. J., Rozenberg G. (Eds.), Proc. 6th Int. Workshop on Theory and Application of Graph Transformation (TAGT '98), LNCS 1764, Springer, Berlin 1999, s. 252÷266.

[BMST99] Bardohl R., Minas M., Schürr A., Taentzer G.: *Application of graph transformation to visual languages,* in Ehrig H., Engels G., Kreowski H. J., Rozenberg G. (Eds.), Handbook of Graph Grammars and Computing by Graph Transformation, vol.2: Applications, Languages and Tools, World Scientific, 1999, s. 105÷180.

[Bart79] Bartini C.: *Rewriting Systems as a Tool for Relational Data Base Design*, LNCS 73, 1979, s. 139÷154.

[BaCo86] Bauderon M., Courcelle B.: *An algebraic formalism for graphs*, in Franchi-Zannettacci P. (Ed.), CAAP'86, LNCS 214, 1986, s. 74÷84.

[BaCo87] Bauderon M., Courcelle B.: *Graph expressions and graph rewriting*, Mathematical Systems Theory 20, 1987, s. 83÷127.

[BGGP11] Bhatt M., Gajek S., Grabska E., Palacz W.: *Artefactual Reasoning in a Hypergraph-Based CAD System*, in Burduk R. et al. (Eds.), Computer Recognition Systems 4, AISC 95, Springer, 2011, s. 471÷478.

[BoFH85] Boehm P., Fonio H., Habel A.: *Amalgamation of Graph Transformations with Applications to Synchronization*, LNCS 185, 1985, s. 267÷285.

[BoGr95] Borkowski A., Grabska E.: *Representing Designs by Composition Graphs,* IABSE REPORTS 72, 1995, s. 27÷36.

[BoGH99] Borkowski A., Grabska E., Hliniak G.: *Function-structure computer-aided design model*, Machine GRAPHICS & VISION 9, 1999, s. 367÷383.

[BoSz01] Borkowski A., Szuba J.: *Graph Transformation in Architectural Design*, Computer Assisted Mechanics and Engineering Science 3, 2001, s. 109÷119.

[BGNS03] Borkowski A., Grabska E., Nikodem P., Strug B.: *Searching for Innovative Structural Layouts by Means of Graph Grammars and Evolutionary Optimization*, in Proc. 2nd Int. Structural Eng. And Constr. Conf, Rome 2003, s. 475÷480.

[CADE11] *CADET,* Home Page- http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cadet/ftp/docs-/CADET.html, 2011.

[CaCK98] Campbell M., Cagan J., Kotovsky K.: *A-Design: Theory and implementation of an adaptive agent-based method of conceptual design*, in Gero J. S., Sudweeks F., (Eds.), Artificial Intelligence in Design'98, Kluwer, Dordrecht 1998, s. 579÷598.

[Cana97] Canamero D.: *Modeling Motivations and Emotions as a Basis for Intelligent Behavior*, Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, California, USA, 1997, s. 148÷155.

[CaMo83] Castellani I., Montanari U.: *Graph Grammars for Distributed Systems*, LNCS 153, 1983, s. 20÷38.

[ChHY97] Chang S. K., Hua W., Yoo C. W.: *Visual Abstraction in the Visual Design Process*, Proc. of SEKE'97, Madrid, Spain, 1997, s. 332÷340.

[Chom65] Chomsky N.: *Aspects of Theory of Syntax*, MIT Press, Cambridge 1965.

[CLHK05] Choset H., Lynch K. M., Hutchinson S., Kantor G., Burgard W., Kavraki L. E., Thrun S.: *Principles of Robot Motion: Theory, Algorithms and Implementations*, MIT Press, 2005.

[CLRD03] Cormen T. H., Leiserson C. E, Rivest R. L, Diks K.: *Wprowadzenie do algorytmów*, WNT,Warszawa 2003.

[Cour88] Courcelle, B.: *Some applications of logic, of universal algebra, and of category theory to the theory of graph transformations*, Bull. EATCS 36, 1988, s. 161÷218.

[CoSm98] Cox P. T., Smedley T.: *LSD: A Logic-Based Visual Language for Designing Structured Objects*, Journal of Visual Languages and Computing 9, 1998, s. 509÷534.

[CoSm00] Cox P. T., Smedley T.: *A Formal Model for Parametrised Solids in a Visual Design Language*, Journal of Visual Languages and Computing 11, 2000, s. 687÷710.

[CRRB90] Coyne R. D., Rosenman M. A., Radford A. D., Balachandran M., Gero J. S.: *Knowledge-based Design System*, Addison-Wesley, Sydney 1990.

[CDKP94] Csuhaj-Varju E., Dassow J., Kelemen J., Paun Gh.: *Grammar systems. A grammatical approach to distribution and cooperation*, Topics in Computer Mathematics 8. Gordon and Breach Science Publishers, Yverdon 1994.

[DoGr01] Do E. Y., Gross M. D.: *Thinking with Diagrams in Architectural Design*, Artificial Intelligence Review, 15, Kluwer Academic Publishers, 2001, s. 135÷149.

[DrHP00] Drewes F., Hoffmann B., Plump D.: *Hierarchical Graph Transformation*, in Turyn J. (Ed.), Proc. of FOSSACS 2000, LNCS 1784, 2000, s. 98÷113.

[Dzer02] Dzeroski S.: *Relational reinforcement learning for agents in worlds with objects*, Proceedings of the Symposium on Adaptive Agents and Multi-Agent Systems (AISB'02), 2002, s. 1÷8.

[EaTe08] Eastman C., Teicholz P., et al.: *BIM Handbook. A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, Wiley, Hoboken, NJ 2008.

[Ehri83] Ehrig H.: *Aspects of Concurrency in Graph Grammars*, LNCS 153, 1983, s. 58÷81.

[EEKR99] Ehrig H., Engels G., Kreowski H. J., Rozenberg G. (Eds.): *Handbook of Graph Grammars and Computing by Graph Transformation*, vol.2: Applications, Languages and Tools, World Scientific, 1999.

[EhKr80] Ehrig H., Kreowski H. J.: *Applications of Graph Grammar Theory to Consistency, Synchronization and Scheduling in Data Base Systems*, Inform. Syst. 5, 1980, s. 225÷238.

[EhLo93] Ehrig H., Löwe M.: *Categorical principles, techniques and results for high-level-replacement systems in computer science*, Applied Categorical Structures 1, 1993, s. 21÷50.

[ErRT99] Ermel C., Rudolf M., Taentzer G.: *The AGG approach: Language and tool environment*, in Ehrig H., Engels G., Kreowski H. J., Rozenberg G. (Eds.), Handbook of Graph Grammars and Computing by Graph Transformation, vol.2: Applications, Languages and Tools, World Scientific, 1999, s. 551÷603.

[FHMV95] Fagin R., Halpern J. Y., Moses Y., Vardi M. Y.: *Reasoning About Knowledge*, MIT Press, Cambridge 1995.

[Flas89] Flasiński M.: *Characteristics of edNLC-Graph Grammars for Syntactic Pattern Recognition*, Computer Vision, Graphics and Image Processing 47, Academic Press, New York, USA, 1989, s. 1÷21.

[Flas95] Flasiński M.: *Use of Graph Grammars for the Description of Mechanical Parts*, Computer-Aided Design 27, Butterworth-Heinemann/ Elsevier Science, Oxford, United Kingdom, 1995, s. 403÷433.

[Flas07] Flasiński M.: *Inference of Parsable Graph Grammars for Syntactic Pattern Recognition***,** Fundamenta Informaticae 80, IOS Press, Amsterdam-Berlin-Oxford 2007, s. 379÷413.

[Flem87] Flemming U.: *More than the Sum of Parts: The Grammar of Queen Anne Houses*, Environment and Planning B: Planning and Design 14, 1987, s. 323÷350.

[FlWo95] Flemming U., Woodbury R.: *Software Environment to support the Early phases in building Design (SEED): Overview*, J. Arch. Engineering 147, 1995.

[FuKS74] Fu K. S.: *Syntactic Methods in Pattern Recognition*, Academic Press, New York 1974.

[Fuja10] *FUJABA –From UML to Java and Back Again*, 2010, www.fujaba.de.

[Furt79] Furtado A. L.: *Transformation of Data Base Structures*, LNCS 73, 1979, s. 224÷236.

[GaPS90] Gaborit P., Potet A., Sayettat C.: *Semantics and validation procedures of a multi-modal logic for formalization of multi-agent universes*, Proceedings of the Ninth European Conference on Artificial Intelligence, Stockholm, Sweden, Pitman Publishing, 1990, s. 289÷291.

[Gaje11] Gajek S.: *Projektowanie koncepcyjne wspomagane komputerowo z użyciem grafowych struktur danych i formuł logicznych*, WFAIS, UJ, praca doktorska w przygotowaniu.

[Gara84] Garavaglia S.: *PROLOG*, Harper and Row, New York 1984.

[Gips99] Gips J.: *Computer Implementation of Shape Grammars*, in Workshop on Shape Computation, MIT, 1999.

[Gold89] Goldberg D. E.: *Genetic Algorithms in Optimization and Machine Learning*, Addison-Wesley, New York 1989.

[GoTh78] Gonzalez R. C., Thomason M. G.: *Syntactic Pattern Recognition – An Introduction*, Addison-Wesley, Reading, Mass., 1978.

[GoCo96] Gooday J. M., Cohn A. G.: *Using spatial logic to describe visual languages*, Proceedings of the Int. Workshop on Theory of Visual Languages, Italy, 1996.

[Gott83] Göttler H.: *Attributed Graph Grammars for Graphics*, LNCS 153, 1983, s. 130÷142.

[GoGN90] Göttler H., Günther J., Nieskens G.: *Use Graph Grammars to Design CAD-Systems*, in Rozenberg G. (Ed.), Graph Grammars and Their Application to Computer Science, LNCS 532, 1990, s. 396÷409.

[Grab93] Grabska E.: *Theoretical Concepts of Graphical Modeling, Part two: CP-graph Grammars and Languages*, Machine Graphics & Vision 2, 1993, s. 149÷178.

[Grab94] Grabska E.: *Graphs and designing*, Lecture Notes in Computer Science 776, 1994.

[GrBo97] Grabska E., Borkowski A.: *Generating floor layouts by means of composite representation*, in Proceedings of the ECCE Symposium on Computers in the Practice of Building and Civil Engineering, Lahti, 1997, s. 154÷158.

[Grab03] Grabska E.: *Design and reasoning with diagrams,* Machine Graphics & Vision 12, 2003, s. 5÷16.

[GrŚP03] Grabska E., Ślusarczyk G., Papiernik K.: *Interpretation of Objects Represented by Hierarchical Graphs*, KOSYR'2003, Wrocław 2003, s. 287÷293.

[GrŚG05] Grabska E., Ślusarczyk G., Grześ P.: *Dynamic Design with the Use of Intelligent Agents*, Proceedings of the 4th International Conference on Computer Recognition Systems CORES'05, Springer, 2005, s. 827÷834.

[GGKŚ06] Grabska E., Grzesiak-Kopeć K., Ślusarczyk G.: *Visual Creative Design with the Assistance of Curious Agents*, in Baker-Plummer D., et al., (Eds.), Diagrammatic Representation on Inference: Proceedings of the 4th International Conference Diagrams, LNCS 4045, 2006, s. 218÷220.

[Grab07] Grabska E.: *Projektowanie wizualne wspomagane komputerem*, EXIT, Warszawa 2007.

[GLŁŚ07] Grabska E., Lembas J., Łachwa A., Ślusarczyk G., Grzesiak-Kopeć K.: *Hierarchical Layout Hypergraph Operations and Diagrammatic Reasoning*, Machine Graphic & Vision 16, 2007, s. 23÷38.

[GrŚL08] Grabska E., Ślusarczyk G., Le T. L.: *Visual Design and Reasoning with the Use of Hypergraph Transformations*, Proceedings of the Seventh International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2008), Electronic Communications of the EASST 10, 2008, s. 305÷318.

[GSŚG08] Grabska E., Strug B., Ślusarczyk G., Grabski W.: *Grammar-Based Distributed Design*, in Rutkowski L. et al. (Eds.), Computational Intelligence: Methods and Applications, EXIT, Warszawa 2008, s. 493÷502.

[GBPG09] Grabska E., Borkowski A., Palacz W., Gajek S.: *Hypergraph System Supporting Design and Reasoning*, EG-ICE International Workshop, Berlin 2009.

[GrSŚ09] Grabska E., Strug B., Ślusarczyk G.: *A Multiagent Distributed Design System*, in Demazeau Y. et al. (Eds.), 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAAMS 2009), Advances in Intelligent and Soft Computing 55, Springer, 2009, s. 364÷373.

[Grab10] Grabska E.: *Visual design with the use of graph-based data structure*, eWork and eBusiness in Architecture, Engineering and Construction – Menzel & Scherer (Eds.), CRC Press, Taylor & Francis Group, London 2010, s. 209-214.

[GrŚl11] Grabska E., Ślusarczyk G.: *Knowledge and reasoning in design systems*, Automation in Construction, 2011, w druku, doi:10.1016/j.autcon.2011.03.009.

[GrŚ11a] Grabska E., Ślusarczyk G.: *Towards intelligent systems supporting conceptual design*, przyjęte do Proceedings of the ICMMI'11, Springer, 2011.

[GrBr00] Grecu D. L., Brown, D. C.: *Expectation formation in multi-agent design systems*, in Gero J. S. (Ed.), Artificial Intelligence in Design'00, Kluwer, Dordrecht 2000, s. 651÷671.

[Guar97] Guarino N.: *Understanding, Building and Using Ontologies*, International Journal of Human Computer Studies, 46, 1997, s. 293÷310.

[GuOS09] Guarino N., Oberle D., Staab S.: *What is an Ontology?*, in Staab S., Studer R. (Eds.), Handbook on Ontologies, International Handbooks on Information Systems, Springer-Verlag, 2009.

[Gurr98] Gurr C. A.: *On the Isomorphism, or Lack of it, of Representations*, in Marriot K., Meyer B. (Eds.) Visual Language Theory, 1998, s. 288÷301.

[Haar99] Haarslev V.: *A Logic-based Formalism for Reasoning about Visual Representations*, Journal of Visual Languages and Computing 10, 1999, s. 421÷445.

[HaKr87] Habel A., Kreowski H. J.: *Some Structural Aspects of Hypergraph Languages Generated by Hyperedge Replacement*, LNCS 247, Springer-Verlag, 1987, s. 207÷219.

[HRWL83] Hayes-Roth F., Waterman D. A., Lenat D. B.: *Building Expert Systems*, Addison-Wesley, New York 1983.

[HeSt73] Herrlich H., Strecker G.: *Category Theory*, Allyn and Bacon, Boston 1973.

[Hils92] Hils D. D.: *Visual Languages and Computing Survey: Data Flow Visual Programming Languages*, Journal of Visual Languages and Computing 3, 1992, s. 69÷101.

[Hoff83] Hoffmann B.: *Modelling Compiler Generation by Graph Grammars*, LNCS 153, 1983, s. 159÷171.

[InFC08] *Industry Foundation Classes*, *Version 2X3*, 2008, http://www.iai-international-.org/index.html.

[Knig80] Knight T.: *The Generation of Hepplewhite-style Chair-back Designs*, Environment and Planning B, vol. 7, 1980, s. 227÷238.

[Knig04] Knight T.: *Interaction in visual design computing*, Visual and Spatial Reasoning in Design III, Key Centre of Design Computing and Cognition, University of Sydney, 2004, the invited paper.

[Koil04] Koile K.: *An Intelligent Assistant for Conceptual Design*, in Gero J. S. (Ed.), Design Computing and Cognition'04, Kluwer Academic Publishers, Dodrecht 2004, s. 3÷22.

[KoEi81] Koning H., Eizengberg J.: *The Language of the Prairie: Frank Lloyd Wright's Prairie Houses*, Environment and Planning B, vol. 8, 1981, s. 295÷323.

[KrMN02] Kraft B., Meyer, O., Nagl M.: *Graph Technology Support for Conceptual Design in Civil Engineering*, Proceedings of the International Workshop on EG-ICE, Darmstadt, 2002, s. 1÷35.

[Kraf03] Kraft B.: *Conceptual Design Tools for Civil Engineering*, AGTIVE 2003, s. 434÷439.

[KrNa07] Kraft B., Nagl M.: *Visual knowledge specification for conceptual design: Definition and tool support*, Advanced Engineering Informatics 21(1), 2007, s. 67÷83.

[Land97] Lander S. E.: *Issues in multiagent design systems*, IEEE Expert 12, 1997, s. 18÷26.

[LeSE06] Lee G., Sacks R., Eastman C. M.: *Specifying parametric building object behavior (BOB) for a building information modeling system*, Automation in Construction, 15(6), 2006, s. 758-776.

[LeNa84] Lewerentz C., Nagl M.: *A Formal Specification Language for Software Systems Defined by Graph Grammars*, Proc. WG'84, Workshop on Graphtheor. Conc. in Computer Science, Berlin 1984.

[LiMP08] Lifschitz V., Morgenstern L., Plaisted D.: *Knowledge Representation and Classical Logic*, Handbook of Knowledge Representation, van Harmelen F., Lifschitz V., Porter B. (Eds.), Elsevier, 2008.

[Lipm87] Lipmann R.: *An introduction to computing with neural nets*, IEEE ASSP Magazine, 1987.

[Maed03] Maeda J.: *Design by numbers*, 2003, http://dbn.media.mit.edu/.

[MaWi82] Mahr B., Wilharm A.: *Graph Grammars as a Tool for Description in Computer Processed Control*, *A Case Study*, Proc. Graphtheoretic Concepts in Computer Science, Hanser-Verlag, Mtnchen/Wien 1982, s. 165÷176.

[McCa02] McCormack J. P., Cagan J.: *Designing inner hood panels through a shape grammar based framework*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 16*,* 2002, s. 273÷290.

[Meie83] Meier A.: *A Graph-Relational Approach to Geographic Data Bases*, LNCS 153, 1983, s. 245÷254.

[MeMW98] Meyer B., Marriot K., Wittenburg K.: *A Survey of Visual Language Specification and Recognition*, in Marriot K., Meyer B. (Eds.) Visual Language Theory, 1998, s. 5÷85.

[Mina97] Minas M.: *Diagram editing with hypergraph parser support*, Proceedings of the 13th IEEE Symposium on Visual Languages, IEEE Computer Society Press, 1997, s. 230÷237.

[Mina02] Minas M.: *Concepts and Realization of a Diagram Editor Generator Based on Hypergraph Transformation*, Science of Computer Programming 44, 2002, s. 157÷180.

[MiVi95] Minas M., Viehstaedt G.: *Diagen: A generator for diagram editors providing direct manipulation and execution of diagrams*, in Proc. IEEE Symposium on Visual Languages (VL'95), Los Alamitos, IEEE Computer Society Press, 1995, s. 203÷210.

[Mins75] Minsky M.: *A Framework for representing knowledge*, in Winston P. H. (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York 1975, s. 211÷277.

[Mous04] Moustapha H.: *A Formal Representation for Generation and Transformation in Design*, Generative CAD Systems Symposium (GCAD'04), Carnegie Mellon University, Pittsburgh 2004.

[MyPo94] Myers L., Pohl J.: *ICDM: integrated cooperative decision making-in practice*, Proceedings of the Sixth International Conference on Tools with Artificial Intelligence, 1994, s. 608÷614.

[Nagl73] Nagl M.: *Beziehungen zwischen verschiedenen Klassen von Diagramm-Sprachen*, Arbeitsbericht 6, Institut für Mathematische Maschinen und Datenverarbeitung, Erlangen 1973, s. 72÷93.

[Nagl79] Nagl M.: *Graph-Grammatiken*, Verlag Vieweg und Sohn, Braunschweig/Wiesbaden 1979.

[Nagl87] Nagl M.: *Set theoretic approaches to graph grammars*, in Ehrig H., Nagl M., Rozenberg G., Rosenfeld A. (Eds.), Graph Grammars and Their Application to Computer Science, LNCS 291, 1987, s. 41÷54.

[Nebe88] Nebendahl E.: *Expert Systems*, J. Wiley and Sons, Inc., Berlin 1988.

[Niko01] Nikodem P.: *Applying Genetic Algorithms in Industrial Designing on Example of Chairs Designing*, KOSYR'2001, Wrocław 2001, s. 325÷330.

[NiSt04] Nikodem P., Strug B.: *Graph Transformations in Evolutionary Design*, Artificial Intelligence and Soft Computing – ICAISC'04, LNCS 3070, 2004, s. 456÷461.

[Quil68] Quilian M. R.: *Semantic memory*, in Minsky M. (Ed.), Semantic Information Processing, MIT Press, 1968, s. 216÷270.

[Pada82] Padawitz P.: *Graph Grammars and Operational Semantics*, Theoret. Comp. Sci. 19, 1982, s. 117÷141.

[PaSa99] Paun Gh., Salomaa A. (Eds.): *Grammatical models of multi-agent systems*. Gordon and Breach, Amsterdam 1999.

[Pavl77] Pavlidis T.: *Structural Pattern Recognition*, Springer-Verlag, New York 1977.

[Petr62] Petri C. A.: *Kommunikation mit Automaten*. PhD Thesis, Fakultät für Mathematik und Physik, Technische Hochschule Darmstadt, Darmstadt, Germany, 1962.

[PfRo69] Pfaltz J. L., Rosenfeld A.: *Web Grammars*, in Walker D. E., Norton L. M. (Eds.), Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington, DC, 1969, s. 609÷620.

[Poko03] Pokojski J. (Ed.): *Zastosowanie metody case-based reasoning w projektowaniu maszyn*, WNT, Warszawa 2003.

[Prat71] Pratt T.: *Pair Graphs, Graph Languages and String-to-Graph Translations*, Jour. Comp. Syst. Sci. 5, 1971, s. 560÷595.

[PROG08] *PROGRES – PROgrammed Graph Rewriting System*, 2008, http://www-i3.informatik.rwth-aachen.de/research/projects/ progres/.

[RMDG97] Rau-Chaplin A., MacKay-Lyons B., Doucette T., Gajewski J., Hu X., Spierenburg P.: *Graphics support for a World-Wide-Web based architectural design service*, Computer Networks and ISDN Systems 29, 1997, s. 1611÷1623.

[RCSm97] Rau-Chaplin A., Smedley T.: *A Graphical Language for Generating Architectural Forms*, in Proceedings of the 13th International IEEE Symposium on Visual Languages, IEEE Computer Society Press, 1997, s. 260÷267.

[ReSc97] Rekers J., Schürr A.: *Defining and parsing visual languages with layered graph grammars*, Journal of Visual Languages and Computing 8, 1997, s. 27÷55.

[RMRF96] Robbins J. E., Morley D. J., Redmiles D. F., Filatov V., Kononov D.: *Visual Language Features Supporting Human-Human and Human-Computer Communication*, Proceedings of the 12th IEEE Symposium on Visual Languages, IEEE Computer Society Press, 1996, s. 247÷254.

[Rose76] Rosenfeld A. (Ed.): *Digital Picture Analysis*, Springer-Verlag, New York 1976.

[RBPE97] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorenson W.: *Object-Oriented Modeling and Design*, Prentice-Hall, London 1997.

[Saun01] Saunders R.: *Curious Design Agents and Artificial Creativity*, Ph.D. Thesis, Faculty of Architecture, The University of Sydney, 2001.

[Schn70] Schneider H. J.: *Chomsky-languages for multidimensional input-media*, International Computing Symposium, Amsterdam: North-Holland, 1970, s. 599÷608.

[Schn75] Schneider H. J.: *Syntax-Directed Description of Incremental Compilers*, LNCS 26, 1975, s. 192÷201.

[Schn79] Schneider H. J.: *Conceptual Data Base Description Using Graph Grammars*, Applied Comp. Sci. 13, Mtnchen, Hanser-Verlag, 1979, s. 77÷98.

[Schu91] Schürr A.: *PROGRESS: A VHL-language based on graph grammars*, Proceedings of the 4th Int. Workshop on Graph-Grammars and Their Applications, LNCS 532, 1991, s. 641÷659.

[ScWZ95] Schürr A., Winter A., Zündorf A.: *Graph grammar engineering with PROGRES*, in Schäfer W., Botella P. (Eds.), Proc. of the 5th European Software Engineering Conference (ESEC95), LNCS 989, 1995, s. 219÷234.

[Shaw69] Shaw A. C.: *The Formal Picture Description Scheme as a Basis for Picture Processing Systems*, Inf. Control 14, 1969, s. 9÷52.

[Sigf96] Sigfried S.: *Understanding Object-Oriented Software Engineering*, IEEE Press, New Jersey 1996.

[Spra06] Sprawozdanie z grantu nr. 5T07F00225, *Transformacje grafowe w konstrukcji systemów wnioskowania diagramowego*, Kraków 2006.

[Stas08] Stasko J.: *Visualization for Information Exploration and Analysis*, IEEE Symposium on Visual Languages and Human-Centric Computing, Germany, 2008.

[StGi72] Stiny G., Gips J.: *Shape grammars and the generative specification of painting and sculpture*, in Freiman C. V. (Ed.), Information Processing 71, North Holland, 1972, s. 1460÷1465.

[StMi78] Stiny G., Mitchell W. J.: *The Palladian grammar*, Environment and Planning B 5, 1978, s. 5÷18.

[SuGP00] Suwa M., Gero J. S., Purcell T.: *Unexpected Discoveries and S-invention of Design Requirements: Important Vehicles for a Design Process*, Design Studies 21, 2000, s. 539÷567.

[Szub05] Szuba J.: *Graphs and Graph Transformations in Design in Engineering*, Ph.D. Thesis, PAS, Warszawa 2005.

[SzSB02] Szuba J., Schürr A., Borkowski A.: *GraCAD – Graph-Based Tool for Conceptual Design,* in Corradini A., Ehrig H., Kreowski H. J., Rozenberg G. (Eds.), First International Conference on Graph Transformation (ICGT 2002), LNCS 2505, Springer-Verlag, Berlin 2002, s. 363÷377.

[Ślus99] Ślusarczyk G.: *Operacje i transformacje grafowe w projektowaniu graficznym*, Ph.D. Thesis, IPPT, Warszawa 1999.

[Ślus03] Ślusarczyk G.: *Hierarchical hypergraph transformations in engineering design*, Journal of Applied Computer Science, 11, 2003, s. 67÷82.

[Ślus04] Ślusarczyk G.: *Heuristic Methods and Hierarchical Graph Grammars in Design*, Visual and Spatial Reasoning in Design III, Key Centre of Design Computing and Cognition, University of Sydney, 2004, s. 45÷66.

[Ślus08] Ślusarczyk G.: *A grammar-based multiagent system in dynamic design*, AIEDAM 22(2), 2008, s. 129÷145.

[TaFl91] Tadeusiewicz R., Flasiński M.: *Rozpoznawanie obrazów*, Państwowe Wydawnictwo Naukowe PWN, Warszawa 1991.

[ThBF05] Thrun S., Burgard W., Fox D.: *Probabilistic Robotics*, MIT Press, 2005.

[Topp98] Toppano E.: *Producing designs of physical systems through model transmutations*, Applications of AI in Engineering XIII, Comp. Mechanics Publications, 1998.

[TrSp95] Traverso P., Spalazzi L.: *A logic for acting, sensing and planning*, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,Montreal, Canada, 1995, s. 1941÷1949.

[TYHT90] Tsuda K., Yoshitaka A., Hirakawa M., Tanaka M., Ichikawa T.: *Iconicbrowser: An iconic retrieval system for object-oriented databases*, Journal of Visual Languages and Computing 1, 1990, s. 59÷76.

[WaLe93] Wang D., Lee J. R.: *Visual Reasoning: its Formal Semantics and Applications*, Journal of Visual Languages and Computing 4, 1993, s. 327÷356.

[WaZe98] Wang D., Zeevat H.: *A syntax directed approach to picture semantics*, in Marriot K., Meyer B. (Eds.) Visual Language Theory, 1998, s. 307÷324.

[Wass89] Wasserman P.D.: *Neural Computing*, Theory and Practice, Van Nostrand, New York 1989.

[Witt88] Wittgenstein L.: *Philosophical Investigations*, Basil Blackwell, 1988.

[Wood95] Wooldridge M. J.: *This is MYWORLD: the logic of an agent oriented DAI testbed*, in Wooldridge M., Jennings N. R. (Eds.), Intelligent Agents: Proceedings of ECAI'94 Workshop on Agent Theories, Architectures and Languages LNAI 890, Springer-Verlag, 1995, s. 160÷178.

[Wood99] Wooldridge M. J.: *Intelligent Agents,* in Weiss G. (Ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge, MA, 1999, s. 27÷77.

[WoLo00] Wooldridge M. J., Lomuscio A.: *Multi-Agent VSK Logic*, in Proceedings of the Seventh European Workshop on Logics in Artificial Intelligence (JELIAI-2000), Springer-Verlag, 2000.

[VoBA97] Voß A.: *Case design specialists in FABEL*, in Maher M. L., Perl P. (Eds.), Issues and Applications of Case-Based Design, Erlbaum, Hillsdale, NJ 1997, s. 301-336.

[Yake00] Yakeley M. W.: *Digitally Mediated Design: Using computer programming to develop a personal design process*, Ph.D Thesis, Department of Architecture, Cambridge, MA, MIT, 2000.

[Yurc09] Yurchyshyna A.: *Modélisation du contrôle de conformité en construction: une approche ontologique,* Ph.D. Thesis, Uniwersytet Nice-Sophia Antipolis, 2009.

[Zara09] Zaraś T.: *Aranżacja wnętrz z wykorzystaniem systemu agentowego sterowanego gramatykami*, praca magisterska, UJ, Kraków 2009.

PROJEKTOWANIE WIZUALNE Z WYKORZYSTANIEM HIERARCHICZNYCH HIPERGRAFÓW

Streszczenie

W rozprawie jest przedstawiony formalny model wizualnego systemu projektowego wykorzystującego wiedzę projektową do wspomagania projektanta na etapie projektowania koncepcyjnego. Dialog między projektantem a systemem odbywa się za pomocą języka wizualnego, którego elementami są diagramy projektowe reprezentujące tworzone rozwiązania. Jednoznaczne wykorzystanie diagramowego języka projektowania do rozwiązywania problemów projektowych jest możliwe dzięki zdefiniowanej tu konceptualizacji dziedziny projektowania wizualnego. Stanowi ona specyfikację pojęć z tej dziedziny, ich taksonomii oraz zachodzących między nimi relacji. Do przekształcania struktur projektów złożonych z obiektów określonych poprzez konceptualizację projektową służą wprowadzone tu syntaktyczne reguły projektowe składające się na system generacyjny. System ten pozwala otrzymać zbiór struktur projektów stanowiących syntaktykę diagramowego języka projektowania. Wyspecyfikowana w pracy realizacja syntaktyki języka projektowania, która nadaje znaczenie geometryczne obiektom projektowym i zachodzącym między nimi relacjom, określa semantykę diagramowego języka projektowania umożliwiając w ten sposób otrzymywanie diagramów reprezentujących projektowane artefakty.

Wewnętrzną komputerową reprezentację diagramów stanowią wprowadzone tu wartościowane atrybutowane hierarchiczne hipergrafy rozmieszczenia. Reprezentacja ta jest łatwa do automatycznego przetwarzania, a jednocześnie pozwala przechowywać zarówno syntaktyczną, jak i semantyczną wiedzę projektową. Hierarchiczne hipergrafy rozmieszczenia składają się z hiperkrawędzi reprezentujących zarówno komponenty diagramów, jak i wieloargumentowe relacje między nimi. Atrybuty przypisane elementom hipergrafów specyfikują własności obiektów projektowych reprezentowanych przez te elementy. Możliwość zagnieżdżania hipergrafów w hierarchicznych hiperkrawędziach reprezentujących grupy komponentów diagramu pozwala projektantowi rozważać projekt na wybranym poziomie szczegółowości, a także wyrażać relacje zachodzące pomiędzy fragmentami projektu znajdującymi się na

różnych poziomach hierarchii. Do transformowania rozważanej reprezentacji wewnętrznej diagramów są wykorzystywane zarówno zdefiniowane tu hierarchiczne hipergrafowe gramatyki rozmieszczenia będące narzędziem generacyjnym umożliwiającym tworzenie projektów zgodnych z narzuconymi kryteriami, jak i wyspecyfikowane operacje na hierarchicznych hipergrafach rozmieszczenia pozwalające na efektywne dokonywanie modyfikacji odpowiadających akcjom projektowym.

Przedstawiony w rozprawie logiczny model wnioskowania o projektach wykorzystuje wewnętrzną reprezentację diagramów projektowych i język logiczny pierwszego rzędu. Hierarchiczne hipergrafy rozmieszczenia są transformowane do postaci formuł logicznych opisujących generowane rozwiązania. Syntaktyka tych formuł bazuje na wprowadzonej wielorodzajowej sygnaturze, a wyspecyfikowane ontologiczne dopasowanie symboli sygnatury do pojęć konceptualizacji projektowej umożliwia jednoznaczne przedstawienie wiedzy dotyczącej obiektów projektowych reprezentowanych przez komponenty diagramów. Semantyka formuł logicznych jest określana za pomocą interpretacji definiowanej poprzez strukturę relacyjną przypisującą atomy hipergrafów i wartości ich atrybutów elementom składowym formuł. Interpretacja ta pozwala na pozyskiwanie wiedzy projektowej dotyczącej diagramów i jej aktualizację podczas wykonywania kolejnych akcji projektowych na diagramach. Porównywanie formuł logicznych, opisujących diagramy, z formułami logicznymi pierwszego rzędu reprezentującymi ograniczenia i wymagania projektowe umożliwia sprawdzanie poprawności rozwiązań projektowych względem zadanych kryteriów.

Zostało zaproponowane wyposażenie systemu wspomagania projektowania wizualnego w zbiór agentów projektowych będących inteligentnymi asystentami wspomagającymi projektanta. Agenci są wyspecjalizowanymi, współpracującymi ze sobą modułami, które autonomicznie rozwiązują powierzone sobie podzadania, operując na hierarchicznej hipergrafowej reprezentacji projektów. Zachowanie agentów jest modelowane za pomocą zbioru reguł hierarchicznych hipergrafowych gramatyk rozmieszczenia i bazy wiedzy zawierającej fakty dotyczące rozwiązywanego zadania. Zdefiniowany tu język hipergrafowy, którego elementy są generowane przez agentów projektowych, określa zbiór możliwych stanów świata. Zaprezentowany semantyczny model wieloagentowego systemu projektowego umożliwia agentom ocenę stanów świata względem predykatów opisujących wymagania projektowe z wykorzystaniem logicznego mechanizmu wnioskowania bazującego na modalnym rachunku predykatów.

Zostały omówione dwa systemy komputerowe wspomagające projektowanie koncepcyjne, które zostały zaimplementowane w Zakładzie Projektowania i Grafiki Komputerowej UJ w sposób zgodny z założeniami przedstawionymi w zaprezentowanym w pracy modelu teoretycznym. Umożliwiają one szybką generację i modyfikację wstępnych rozwiązań projektowych oraz wnioskowanie na temat ich poprawności. Wykorzystanie zaproponowanego

formalnego modelu komputerowego wspomagania projektowania wizualnego z wykorzystaniem wiedzy projektowej jest tu przedstawione na przykładach projektowania rozkładów i aranżacji pomieszczeń, krzeseł i kratowych wież przesyłowych.

VISUAL DESIGN WITH THE USE OF HIERARCHICAL HYPERGRAPHS

Abstract

A formal model of the visual design system, which uses design knowledge to support the designer during the conceptual design phase, is presented. A dialogue between the designer and the system is performed by means of a visual language composed of diagrams representing created solutions. The unambiguous use of the diagrammatic design language for solving design problems is assured by a conceptualization of the visual design domain defined here. It specifies concepts of this domain, their taxonomy and relationship between them. To transform design structures composed of objects determined by the design conceptualization, syntactic design rules that form a generative system introduced here, are used. This system allows the designer to obtain a set of design structures forming syntax of the diagrammatic design language. A specified realization of the design language syntax, which gives geometric meaning to design objects and relations between them, defines the semantics of the diagrammatic design language allowing to receive diagrams representing designed artifacts.

Diagrams are internally represented in the system by evaluated attributed hierarchical layout hypergraphs, which are introduced here. This representation is easy for automatic processing and at the same time allows for encoding both syntactic and semantic design knowledge. Hierarchical layout hypergraphs consist of hyperedges representing both diagram components and multi-argument relations between them. Attributes assigned to hypergraph elements specify properties of design objects represented by these elements. The possibility of nesting hypergraphs in hierarchical hyperedges representing groups of diagram components allows the designer to consider the project at the chosen level of detail and to express the relationships between fragments of the project placed on different hierarchy levels. To transform the considered internal representation of diagrams, both hierarchical hypergraph layout grammars defined here, which constitute a generative tool for creating projects consistent with imposed design criteria, and specified operations on hierarchical

layout hypergraphs, which enable their effective modifications corresponding to the design actions, are used.

The presented logical model of reasoning about projects uses the internal representation of diagrams and the first-order logic language. Hierarchical layout hypergraphs are transformed into logical formulas describing generated solutions. The syntax of these formulas is based on the introduced multi-sorted signature, while the specified ontological commitment, which maps signature symbols to concepts of the design conceptualization, allows an unambiguous presentation of knowledge concerning the design objects represented by the diagram components. The semantics of logical formulas is specified using the interpretation defined by the relational structure assigning hypergraph elements and values of their attributes to constituent elements of formulas. This interpretation allows for the acquisition of knowledge concerning design diagrams, and its updating whilst the design actions on diagrams are performed. By comparing the logical formulas describing diagrams to first-order logic formulas representing the constraints and design requirements, design solutions are validating against the given criteria.

The endowment of the visual design system with a set of design agents being intelligent assistants aiding the designer has been proposed. Agents are specialized, co-operating modules, which independently solve assigned subtasks by manipulating the hierarchical hypergraph representations of projects. The behavior of agents is modelled by a set of hierarchical layout hypergraph grammar rules and a knowledge base containing facts about a problem to be solved. The hypergraph language defined here, the elements of which are generated by design agents, defines a set of possible states of the world. The presented semantic model of the multi-agent design system allows agents to assess states of the world, according to predicates describing the design requirements, using the logical inference mechanism based on the modal predicate calculus.

Two computer systems supporting the conceptual design, which have been implemented in the Department of Design and Computer Graphics at the Jagiellonian University, in a manner consistent with the objectives outlined in the presented theoretical model, are discussed. They rapidly generate and modify early design solutions, and reason about their correctness. The application of the proposed formal model of computer-aided visual design with the use of design knowledge is presented on examples of designing floor layouts, arrangements of rooms, chairs and transmission towers.

SŁOWNIK WYBRANYCH TERMINÓW

język wizualny – język wykorzystujący obrazy do wyrażania i przekazywania koncepcji

diagram projektowy – suma przetransformowanych kształtów elementarnych reprezentujących obiekty projektowe

diagramowy język projektowania – zbiór diagramów projektowych będących wizualizacją rozwiązań projektowych

konceptualizacja projektowa – specyfikacja pojęć z dziedziny projektowania wizualnego, ich taksonomii oraz zachodzących między nimi relacji

konceptualizacja zadaniowa – wynik przypisania elementów konkretnej dziedziny projektowania pojęciom konceptualizacji projektowej

syntaktyczne reguły projektowe – reguły przekształcania struktur projektów

projektowy system generacyjny – początkowa struktura projektu wraz ze zbiorem syntaktycznych reguł projektowych

syntaktyka diagramowego języka projektowania – zbiór struktur projektów otrzymanych z wykorzystaniem systemu generacyjnego

semantyka diagramowego języka projektowania – znaczenie geometryczne nadawane elementom struktur projektów poprzez realizację syntaktyki diagramowego języka projektowania

hipergraf rozmieszczenia – atrybutowany hierarchiczny hipergraf stanowiący wewnętrzną reprezentację struktury projektu odpowiadającej diagramowi projektowemu

hipergrafowa gramatyka rozmieszczenia – narzędzie umożliwiające generację hipergrafów rozmieszczenia

dopasowanie ontologiczne – odwzorowanie pomiędzy symbolami słownika języka logicznego pierwszego rzędu i pojęciami konceptualizacji projektowej

struktura relacyjna – interpretacja formuł logicznych złożona z określonej dziedziny elementów wraz z ich przypisaniem symbolom tych formuł

agent projektowy – inteligentny agent wyposażony w zorientowaną problemowo wiedzę projektową

wieloagentowy system projektowy – zbiór agentów projektowych stanowiący modularny współbieżny system rozwiązujący zadania projektowe

WYKAZ WYBRANYCH OZNACZEŃ

Diagramowy język wizualny

Hierarchiczne hipergrafy w projektowaniu

Operacje na hierarchicznych hipergrafach

Hierarchiczne gramatyki hipergrafowe

Wnioskowanie z hipergrafowej reprezentacji diagramów

Agenci w projektowaniu

INFORMATION FOR AUTHORS

The journal *STUDIA INFORMATICA* publishes both fundamental and applied Memoirs and Notes in the field of informatics. The Editors' aim is to provide an active forum for disseminating the original results of theoretical research and applications practice of informatics understood as a discipline focused on the investigations of laws that rule processes of coding, storing, processing, and transferring of information or data.

Papers are welcome from fields of informatics inclusive of, but not restricted to *Computer Science*, *Engineering*, and *Life and Physical Sciences*.

All manuscripts submitted for publication will be subject to critical review. Acceptability will be judged according to the paper's contribution to the art and science of informatics.

In the first instance, all text should be submitted as hardcopy, conventionally mailed, and for accepted paper accompanying with the electronically readable manuscript to:

Dr. Marcin SKOWRONEK
Institute of Informatics
Silesian University of Technology
ul. Akademicka 16
44-100 Gliwice, Poland
Tel.: +48 32 237-12-15
Fax: +48 32 237-27-33
e-mail: marcin.skowronek@polsl.pl

MANUSCRIPT REQUIREMENTS

All manuscripts should be written in Polish or in English. Manuscript should be typed on one side paper only, and submitted in duplicate. The name and affiliation of each author should be followed by the title of the paper (as brief as possible). An abstract of not more than 50 words is required. The text should be logically divided under numbered headings and subheadings (up to four levels). Each table must have a title and should be cited in the text. Each figure should have a caption and have to be cited in the text. References should be cited with a number in square brackets that corresponds to a proper number in the reference list. The accuracy of the references is the author's responsibility. Abbreviations should be used sparingly and given in full at first mention (e.g. "Central Processing Unit (CPU)"). In case when the manuscript is provided in Polish (English) language, the summary and additional abstract (up to 300 words with reference to the equations, tables and figures) in English (Polish) should be added.

After the paper has been reviewed and accepted for publication, the author has to submit to the Editor a hardcopy and electronic version of the manuscript.

It is strongly recommended to submit the manuscript in a form downloadable from web site http://zti.polsl.pl/makiety/.

To subscribe: *STUDIA INFORMATICA* (PL ISSN 0208-7286) is published by Silesian University of Technology Press (Wydawnictwo Politechniki Śląskiej) ul. Akademicka 5, 44-100 Gliwice, Poland, Tel./Fax +48 32 237-13-81. 2011 annual subscription rate: US$60. Single number price approx. US$10-20 according to the issue volume.

INSTYTUT INFORMATYKI prowadzi:

- ❑ Studia stacjonarne I stopnia (inżynierskie)
- ❑ Studia stacjonarne II stopnia (magisterskie)
- ❑ Studia niestacjonarne I stopnia (inżynierskie)
- ❑ Studia niestacjonarne II stopnia (magisterskie)
- ❑ Studia podyplomowe:
  - *Sieci i systemy komputerowe, bazy danych*
  - *Systemy informacji geograficznej*
  - *Teleinformatyka w transporcie lotniczym*
  - *Technologie internetowe i technologie mobilne*
  - *Metody eksploracji baz danych przedsiębiorstw*
- ❑ Studia doktoranckie

Informacje:

POLITECHNIKA ŚLĄSKA
Instytut Informatyki

44-100 Gliwice, ul. Akademicka 16
tel. (032) 237 24 05; 237 21 51;
fax (032) 237 27 33 (czynny całą dobę)

e-mail: rau2@polsl.pl

http://www.inf.polsl.pl (dydaktyka)