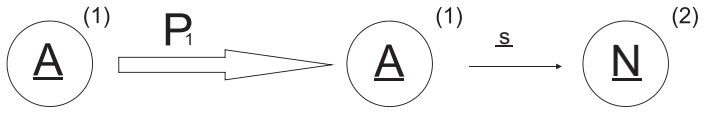
Zaprezentowane poniżej cztery produkcje pozwolą generować grafy, reprezentując aktualny stan alokacji oprogramowania w kontrolowanym systemie rozproszonym. Inicjalny IE-graf alokacji składa się z pojedynczego wierzchołka etykietowanego jako A i indeksowanego wartością 0.

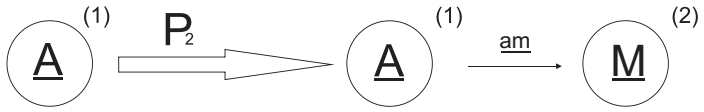
Produkcja P_1 wprowadza do grafu alokacji G nowy wierzchołek reprezentujący węzeł obliczeniowy (etykietowany jako N) i wiąże go z wierzchołkiem reprezentującym alokator (indeks 0).



Transformacja osadzenia dla tej produkcji jest następująca:

$$\begin{aligned} C_1 \quad = \quad & \{((s, out, v_L), \{(A, (N, true), s, out), \\ & (N, (N, true), -n, in), (N, (N, true), n, in)\}), \\ & CopyRest)\} \end{aligned}$$

Kiedy chcemy zaalokować oprogramowanie w postaci instancji obiektu OBJ (ze zdefiniowanymi zbiorami żądanych i oferowanych usług – odpowiednio $reqeusted\_services$ i $offered\_services$) musimy go połączyć ze wszystkimi jest modułami pośredniczącymi (ang. *stub*) do jednej z usług żądanych przez alokowany obiekt, pod warunkiem, że są one zaalokowane w węźle obliczeniowym NODE w zadanym węźle obliczeniowym (którego indeks jest podany w zmiennej NODE) powinniśmy zastosować produkcję P_2.



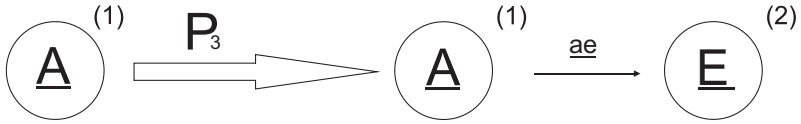
Transformacja osadzenia dla tej produkcji jest następująca:

$$\begin{aligned} C_2 \quad = \quad & \{((s, out, v_L), \{(A, (N, true), s, out), \\ & (M, (N, \pi_1), -a, in)\}), ((ai, out, v_L), \\ & \{(A, (I, true), ai, out), (M, (I, \pi_{2)}, -c, in)\}), \\ & CopyRest)\}, \end{aligned}$$

gdzie:

- $\pi_1 : NODE = name._{CN}$,
- $\pi_2 : service_{CN} \in requested\_services_{OBJ} \wedge Allocated\_in(CN) = NODE.$

Kontynuując alokację obiektu, pamiętamy pomocniczo jego indeks w zmiennej globalnej ALLOBJ i dla każdej z usług $serv$ oferowanych przez alokowany obiekt ($serv \in offered\_services$) wykonujemy produkcję P_3.

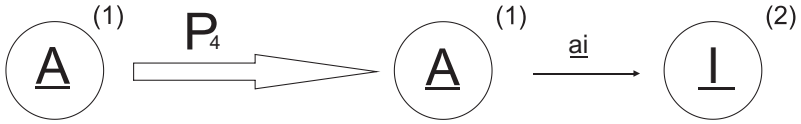


Transformacja osadzenia dla tej produkcji jest następująca:

$$\begin{aligned} C_3 \quad = \quad & \{((am, out, v_L), \{(A, (M, true), am, out), \\ & (E, (M, \pi_3), -b, in)\}), CopyRest)\}, \end{aligned}$$

gdzie $\pi_3 : CN = AO$.

Następnie dla wszystkich usług żądanych albo istnieje już zaalokowany moduł pośredniczący do tej usługi w tym węźle obliczeniowym, i powiązanie wierzchołka M z I zostało wykonane w ramach produkcji P_2 albo musimy wykonać produkcję P_4, która go wygeneruje.



Transformacja osadzenia dla tej produkcji jest następująca:

$$\begin{aligned} C_5 \quad = \quad & \{((ae, out, v_L), \{(A, (E, true), ae, out), \\ & (I, (E, \pi_5), -l, in)\}), ((am, out, v_L), \\ & \{(A, (M, true), am, out), (I, (M, \pi_6), c, in)\}), \\ & CopyRest\}, \end{aligned}$$

gdzie $\pi_5 : service_2 = serv_{CN} \wedge \pi_6 : CN = AO$.

Zwróćmy uwagę, że wierzchołek reprezentujący alokowany obiekt AO nie występuje po lewej ani po prawej stronie produkcji, ale ponieważ istnieje krawędź (etykietowana jako am) pomiędzy wierzchołkiem lewej strony produkcji i AO, to możemy wygenerować krawędź pomiędzy AO i wierzchołkiem wstawianym jako wierzchołek indeksowany liczbą 2 w grafie prawej strony produkcji (etykietowanym etykietą I).

1.2. Algebraiczne transformacje grafowe

Podejście algebraiczne bazuje na konstrukcji $pushout$, która to konstrukcja pozwala formalnie zamodelować sklejanie grafów w kontekście opisu w ramach teorii kategorii. W praktyce stosuje się dwa główne podejścia:

1) *double pushout* (skrót DPO) – wprowadzone przez Ehriga [20] i rozwijane w szczególności przez zespoły związane z uniwersytetami w Berlinie i Pizie,
2) *single pushout* (skrót SPO) – wprowadzone przez Löwe'a [19, 76] i rozwijane równolegle do podejścia DPO.

Z uwagi na fakt, że teoria kategorii operuje na zbiorach i funkcjach (morfizmach) pomiędzy nimi, krawędzie definiowane wcześniej jako $E \subset V \times \Gamma \times V$ będą w definicji grafu (def. 1.4) definiowane jako zbiór abstrakcyjnych bytów zwanych krawędziami oraz funkcji s, ϵ, t.

Definicja 1.4. Grafem uogólnionym nad zbiorem etykiet wierzchołków Σ oraz krawędzi Γ nazywamy dziesiątkę $H = (V, E, \Sigma, \Gamma, \delta, \lambda, s, t, \upsilon, \varepsilon)$, gdzie:

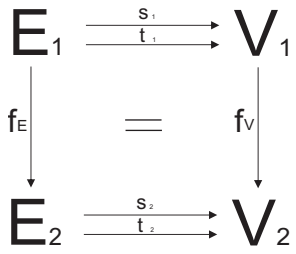
- V jest skończonym, niepustym zbiorem wierzchołków grafu,
- E jest zbiorem krawędzi,
- Σ jest zbiorem etykiet wierzchołkowych,
- Γ jest zbiorem etykiet krawędziowych,
- $\delta : V \longrightarrow \Sigma$ jest funkcją etykietowania wierzchołków,
- $\lambda : E \longrightarrow \Gamma$ jest funkcją etykietowania krawędzi,
- $s : E \longrightarrow V$ wskazuje wierzchołek będący punktem startowym krawędzi $(source)$,
- $t : E \longrightarrow V$ wskazuje wierzchołek będący punktem docelowym krawędzi $(target)$,
- $\nu : V \longrightarrow (NodeAtrib \rightarrow NodeAtribValue)$ jest funkcją atrybutującą wierzchołki,
- $\varepsilon : E \longrightarrow (EdgeAtrib \rightarrow EdgeAtribValue)$ jest funkcją atrybutującą krawędzie.

Istnieje równoważność pomiędzy obiema notacjami, gdyż dla $e = (p, \lambda, q)$ (z def. 1.1) wartości funkcji s i t będą zdefiniowane następująco: $s(e) = p$, $\epsilon(e) = \lambda$ i $t(e) = q$.

Podstawą formalną do wprowadzenia algebraicznych transformacji grafowych są pojęcia morfizmu grafów i kategorii.

Definicja 1.5. Dla danych grafów G_1 i G_2, zdefiniowanych jako $(V_i, E_i, \Sigma, \Gamma, \delta_i, \epsilon_i, s_i, t_i, \upsilon_i, \varepsilon_i)$ dla $i = 1, 2$ morfizm $f : G_1 \longrightarrow G_2$ zdefiniowany jest jako trójka funkcji $f = (f_V : V_1 \to V_2, f_E : E_1 \to E_2, f_R)$ takich, że:

- f_V i f_E zachowują własności funkcji s i t, czyli $f_V \circ s_1 = s_2 \circ f_E$ oraz $f_V \circ t_1 = t_2 \circ f_E$;
- $f_R : (\delta_1, \lambda_1, \upsilon_1, \varepsilon_1) \longrightarrow (\delta_2, \lambda_2, \upsilon_2, \varepsilon_2)$ przypisuje odpowiednie mapowanie dla etykietowań i atrybutowań krawędzi oraz wierzchołków, czyli $F_R(p, \alpha, q, \beta) = (\delta \circ f_V(p), \lambda \circ f_E(\alpha), \nu \circ f_V(q), \varepsilon \circ f_E(\beta))$.



Morfizm f nazywamy różnowartościowym, jeżeli odpowiednio funkcje f_E i f_V są różnowartościowe, i izomorfizmem, jeżeli funkcje f_E i f_V są bijekcjami.

Definicja 1.6. Kategoria $\mathcal{K} = (Obj_K, Mor_K, \circ, id)$ zdefiniowana jest następująco:

- Obj_K jest zbiorem obiektów,
- dla każdej pary $A, B \in Obj_K$ $Mor_K(A, B)$ jest zbiorem morfizmów,
- dla każdej trójki $A, B, C \in Obj_K$ operacja złożenia morfizmów $\circ$ jest też morfizmem czyli $\circ_{(A,B,C)} : Mor_K(B, C) \times Mor_K(A, B) \longrightarrow Mor_K(A, C)$,
- dla każdego $A \in Obj_K$ morfizm identycznościowy $id_A \in Mor_K(A, A)$.

Ponadto obiekty spełniają następujące warunki:

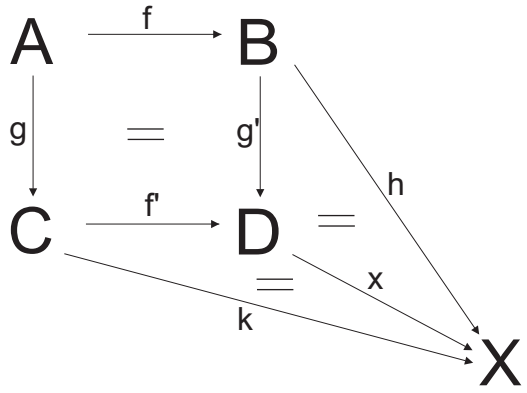
- łączności: dla $A, B, C, D \in Obj_K$ i $f \in Mor_K(A, B)$, $g \in Mor_K(B, C)$, $h \in Mor_K(C, G)$ prawdziwe jest $(h \circ_{(B,C,D)} g) \circ_{(A,B,C)} f = h \circ_{(B,C,D)} (g \circ_{(A,B,C)} f)$,
- tożsamości: dla $A, B \in Obj_K$ i $f \in Mor_K(A, B)$ prawdziwe są wyrażenia $f \circ_{(A,A,B)} id_A = f$ i $id_A \circ_{(A,B,B)} f = f$.

Przyjmujemy też następujące konwencje: $f \in Mor_K(A, B)$ oznaczamy jako $f : A \longrightarrow B$, obiekt A nazywamy dziedziną morfizmu f, a obiekt B jego przeciwdziedziną oraz opuszczamy indeks w operacji kompozycji, gdy wybór kompozycji jednoznacznie wynika z kontekstu. Przykłady zastosowania teorii kategorii do opisu systemów grafowych można znaleźć w [18].

Do wprowadzenia transformacji grafowych potrzebujemy formalizmu, który pozwoli opisać sklejenie grafów, wykorzystując wspólny podgraf.

Definicja 1.7. Dla zadanych morfizmów $f : A \longrightarrow B$ i $g : A \longrightarrow C$ w kategorii $\mathcal{K}$ pushoutem nad f i g nazwiemy trójkę (D, f', g'), gdzie:

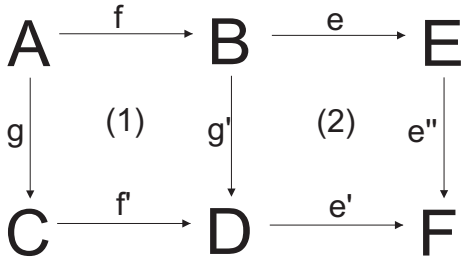
- D jest obiektem pushoutu,
- $f' : C \longrightarrow D$ i $g' : B \longrightarrow D$ są morfizmami takimi że $f' \circ g = g' \circ f$ oraz następujący warunek jest spełniony: dla dowolnego obiektu $X \in Obj_K$ i morfizmów $h : B \longrightarrow X$ i $k : C \longrightarrow X$ istnieje jedyny morfizm $x : D \longrightarrow X$, taki że $x \circ g' = h$ i $x \circ f' = k$.



Dla zdefiniowanej konstrukcji pushoutu można zbadać takie właściwości jak jednoznaczność, składanie i dekompozycja pushoutów. W [18] udowodniono niżej podane twierdzenie.

Twierdzenie 1.1. Dla zadanej kategorii $\mathcal{K}$ prawdziwe są następujące stwierdzenia:

- pushout D jest wyznaczony jednoznacznie przez izomorfizmy,
- składanie i dekompozycja pushoutów też jest pushoutem, czyli dla zaprezentowanego poniżej diagramu prawdziwe są stwierdzenia:
  - złożenie pushoutów: jeżeli (1) i (2) są pushoutami, to $(1) + (2)$ też jest pushoutem.
  - dekompozycja pushoutu: jeżeli (1) i $(1) + (2)$ są pushoutami, to (2) też jest pushoutem.

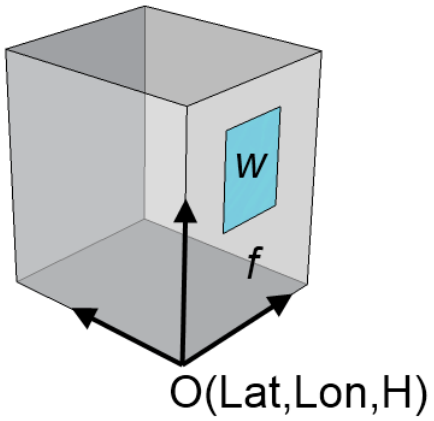


Definicja 1.21. Hipergrafem bryły wypukłej S nazywać będziemy etykietowany i atrybutowany hipergraf $G_S = (V, A \cup H, \Sigma, \Gamma, \delta, \lambda, \nu, \varepsilon)$, gdzie:

- V jest skończonym, niepustym zbiorem wierzchołków grafu reprezentujących płaszczyzny (ściany bryły), którego wierzchołki są jednoznacznie rozróżnialne przez funkcję indeksującą I,
- $E = A \cup H$ jest zbiorem krawędzi, gdzie dla $e \in E$ funkcja $s(e)$ zwraca sekwencję wierzchołków, z których krawędź wychodzi, a funkcja $t(e)$ zwraca zbiór wierzchołków, do których krawędź dochodzi; zbiory A i H mają następującą interpretację: $A \subset V \times V$ oraz $H \subset \bigcup_{i>2} V^i$ (gdzie V^ioznacza i-elementowy iloczyn kartezjański):
  - każda hiperkrawędź $e = \{v_{f_1}, v_{f_2}\} \in A$ reprezentuje dokładnie jedną krawędź bryły S, wspólną dla ścian f_1 i f_2,
  - każda hiperkrawędź $e = \{v_{f_1}, v_{f_2}, \ldots v_{f_k}\} \in H$ $(k > 2)$ reprezentuje dokładnie jeden wierzchołek bryły S wspólny dla ścian $f_1, f_2, \ldots f_k$.

Pozostałe elementy są identyczne jak w definicji 1.3. Rodzinę hipergrafów brył wypukłych będziemy oznaczać jako $\mathcal{H}_{conv}$. □

Rysunek 1.11 reprezentuje prostopadłościan z dodatkowym elementem – oknem, a rysunek 1.12 przedstawia część jej grafowej reprezentacji – linie ciągłe ilustrują krawędzie $e \in A$, a linia przerywana krawędzie należące do $e \in H$.



Rysunek 1.11. Sześcian reprezentujący budynek z oknem na ścianie szczytowej i odpowiadający mu fragment hipergrafu

2. Rozproszone transformacje grafowe – teoria

Przedstawiony w poprzednim rozdziale aparat formalny transformacji grafowych znalazł zastosowanie w wielu dziedzinach, szczególnie tych, które zajmują się budową i weryfikacją formalnego modelu systemu.

Przy praktycznym zastosowaniu tych formalizmów, w szczególności w kontekście dużych wolumenów danych (czyli grafów liczonych w setkach tysięcy węzłów), pojawiają się problemy z efektywnością rozwiązania. Z drugiej jednak strony regularne, dobrze zdefiniowane struktury danych wprost zachęcają do ich analizy w sposób równoległy. Paradygmat przetwarzania rozproszonego zarówno umożliwia kooperację podsystemów geograficznie rozproszonych, jak i podnosi efektywność i niezawodność takich systemów (jeżeli ich dane i funkcjonalności są właściwie zreplikowane). Zauważmy, że z teoretycznego punktu wiedzenia (przy nieograniczonej liczbie procesorów i bez uwzględnienia narzutów komunikacyjnych) model obliczeń równoległych pozwala na efektywne rozwiązanie problemów NP-zupełnych, o ile formalizm opisu problemu wspiera obliczenia równoległe. W szczególności Kreowski i Kuske wykazali [73], że za pomocą formalizmu „graph multiset transformations" [74] można rozwiązać NP-zupełny problem „Hamilton paths" w czasie wielomianowym (czyli $NP_{GraphTransf} = P_{ParallelGraphTransf}$). Niestety w chwili obecnej nie dysponujemy efektywną implementacją wspomagającą formalizm „graph multiset transformations" w czasie wielomianowym, ale powyższe rezultaty teoretyczne powinny nas zachęcić do rozwoju narzędzi wspomagających przetwarzanie równoległe, szczególnie w środowisku rozproszonym.

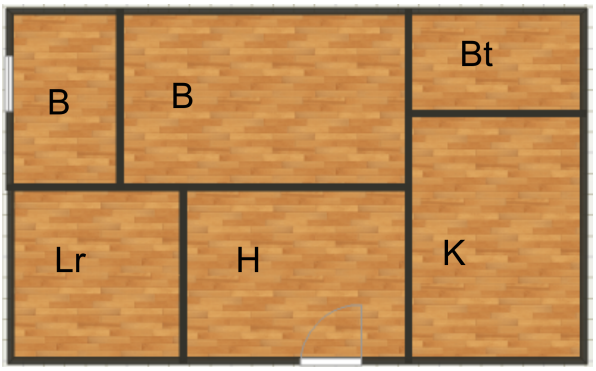
Wprowadzenie współbieżnego dostępu do współdzielonych struktur danych, które mogą być dynamicznie modyfikowane, wprowadza nowe klasy błędów zwanych błędami uwarunkowanymi czasowo oraz zakleszczeniami systemu. W klasycznych rozwiązaniach systemów współbieżnych wprowadza się dodatkowe mechanizmy umożliwiające synchronizację, takie jak: semafory, monitory, mechanizm *rendez-vous* z języka Ada czy synchroniczną wymianę komunikatów. Dokładniej te problemy są omówione w monografii Tanenbauma i Steena [95]. Należy zwrócić jednak uwagę, że poprawne wprowadzenie synchronizacji nie jest zadaniem łatwym,

ponadto wymaga dodatkowego wysiłku i jak każdy projekt programistyczny jest procesem długotrwałym. Na koniec okazuje się, że rozwiązania te są dedykowane dla konkretnego rozwiązania i nawet niewielka modyfikacja może wymagać przeprojektowania całego systemu synchronizacji.

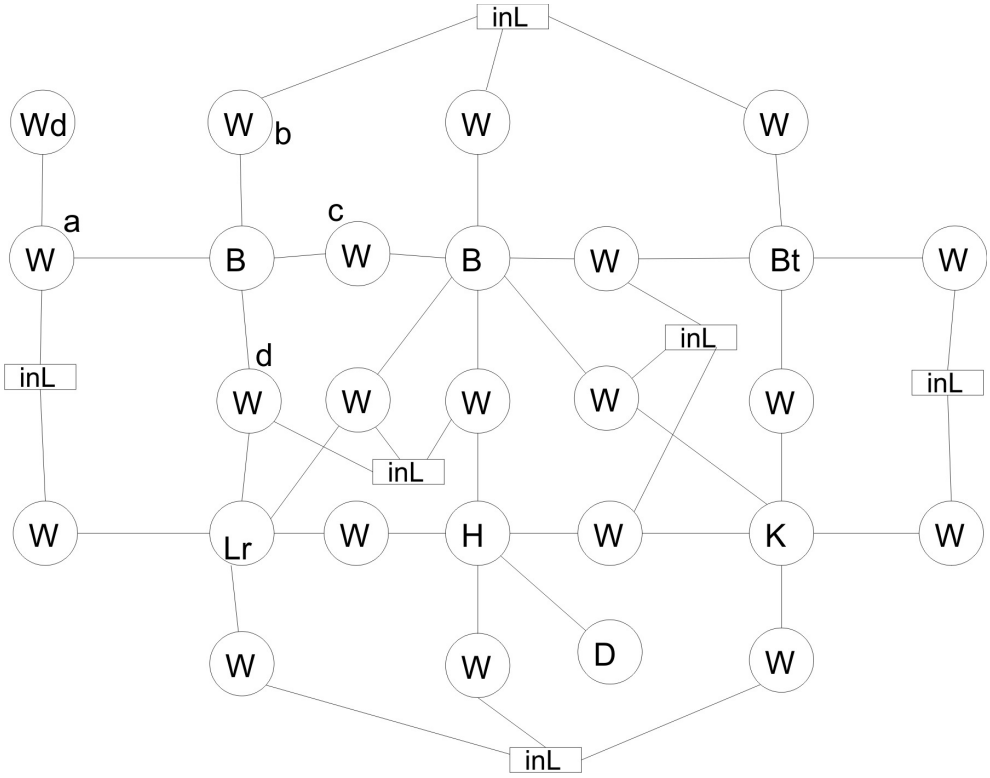
Celowe wydawałoby się opracowanie mechanizmu niejawnej synchronizacji (ang. *implicit synchronization*), pozwalającego na wprowadzanie elementu synchronizacji do mechanizmu współpracy systemów rozproszonych, które byłyby uzależnione jedynie od typu gramatyki, a nie od konkretnych rozwiązań (czyli określonych zbiorów produkcji). Wstępna koncepcja takiego rozwiązania została podana w [49]. Zakładamy podział scentralizowanego grafu G na podgrafy $G_1, ...G_k$ w taki sposób, by można było efektywnie skleić je w jeden scentralizowany graf. Grafy G_i $i = 1...k$ nazywać będziemy częściowymi, gdy będziemy chcieli pokazać ich autonomię, lub komplementarnymi, gdy będziemy chcieli podkreślić aspekt ich współzależności.

Nad każdym z podgrafów częściowych G_i opiekę przejmuje agent G_i, którego zadaniem będzie dokonywanie lokalnych modyfikacji podgrafu oraz nawiązanie współpracy z innymi agentami w celu wykonania transformacji, która nie może być wykonana lokalnie w taki sposób, by po sklejeniu grafów komplementarnych wynik był identyczny jak wynik wykonania pewnej sekwencji transformacji na grafie scentralizowanym. Koncepcja grafów komplementranych jest od kilku lat rozwijana w ramach prowadzonej przez autora grupy badawczej GRADIS. Aparat formalny dla takiego rozwiązania, nazwanego RCG (ang. *Replicated Complementary Graps*), w którym linia podziału pomiędzy grafami komplementarnymi przebiega przez wierzchołki, zostanie opisany w podrozdziale 2.1 (dotyczącym grafów) oraz podrozdziale 2.1.3 (dotyczącym transformacji grafowych). W obu przypadkach przeanalizowana zostanie złożoność obliczeniowa algorytmów wspomagających powyższe koncepcje.

Niezależnie od stworzenia ram formalnych umożliwiających realizację rozproszonych transformacji grafowych do efektywnej pracy systemu opartego na RCG wymagany jest efektywny podział początkowy grafu scentralizowanego na grafy komplementarne według zadanego kryterium podziału. Zadanie to należy do NP-trudnych obliczeniowo, musimy więc przeanalizować możliwość wykorzystania algorytmów heurystycznych, których analizie poświęcony jest podrozdział 2.2. Uzupełnieniem tych rozważań będzie zasygnalizowanie rozwijanej ostatnio (w ramach grupy GRADIS) przez Sędziwego [85] koncepcji tzw. Slashed Graph opartej na podziale grafów przez rozcinanie krawędzi. Na zakończenie rozdziału omówiona będzie koncepcja współpracy różnych typów gramatyk grafowych na bazie sprzężenia (wymuszenia równoległego wykonania) transformacji grafowych w różnych gramatykach opisujących poszczególne aspekty tego samego problemu (patrz podrozdz. 2.4).



(a) Przykład rzutu 2D mieszkania



(b) Hipergraf reprezentujący mieszkanie z rysunku 3.13a

Rysunek 3.13. Graficzna i hipergrafowa reprezentacja układu mieszkania

Do opisania geometrii brył reprezentujących przestrzeń miejską stosuje się dwa układy współrzędnych. Pierwszy z nich, **globalny**, może być dowolnym geodezyjnym układem odniesienia (WGS84, UTM), o którym można założyć, że jest liniowy w modelowanym obszarze. Drugi układ, **lokalny**, definiuje się poprzez powiązanie z określonym punktem bryły S początku tego układu, O_S, oraz poprzez określenie wektorów bazowych $\{\mathbf{e}_i\}$ (np. baza ortonormalna).

W celu łatwego przechodzenia między obydwoma układami współrzędnych z każdym węzłem i hiperkrawędzią hipergrafu $\hat{G}(S)$, oprócz atrybutu określającego lokalne współrzędne, wiążemy dwa atrybuty (niezmiennicze w ramach $\hat{G}(S)$): LocalOrigin określający globalne współrzędne punktu O_S oraz LocalBase zawierający definicję bazy $\{\mathbf{e}_i\}$.

Również w przypadku hipergrafu reprezentacji wewnętrznej wprowadzamy dwa układy współrzędnych. Pierwszy z nich jest lokalnym układem współrzędnych, określonym powyżej (LocalOrigin), natomiast drugi jest sublokalnym układem odniesienia, skojarzonym z kondygnacją budynku, którego początek, wyrażony we współrzędnych lokalnych, przechowywany jest w atrybucie StoreyOrigin. Dla zapewnienia komunikacji między oboma systemami transformacyjnymi, oprócz lokalnego układu współrzędnych (LocalOrigin), współdzielone są również atrybuty: EntityType, którego wartościami mogą być okno, drzwi itp., oraz EntityProperties zawierający zbiór własności specyficznych dla obiektu (w tym współrzędne).

Hipergrafy reprezentujące projektowane obiekty mogą być generowane w wywodach gramatyk hipergrafowych. Gramatyka hipergrafowa złożona jest ze zbioru hiperkrawędzi o terminalnych i nieterminalnych etykietach, zbioru wierzchołków, zbioru produkcji oraz hipergrafu startowego.

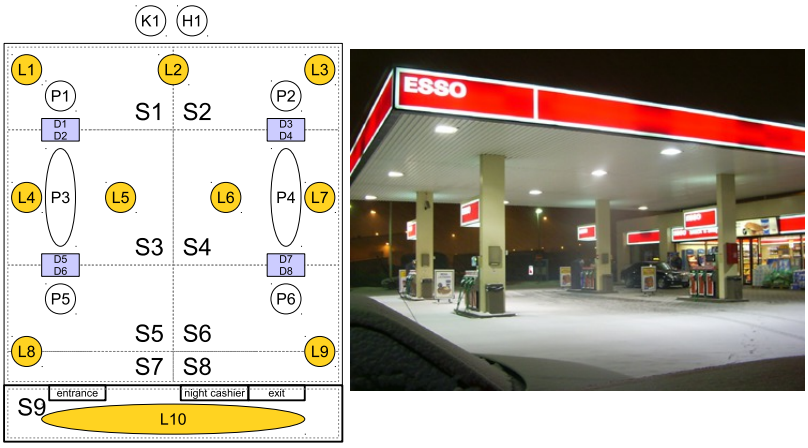
Definicja 3.1. Niech $\hat{\mathscr{H}}$ będzie rodziną instancji etykietowanych, atrybutowanych hipergrafów. Produkcją gramatyki hipergrafowej nazywamy czwórkę postaci $p = (l, r, \pi, eval)$, gdzie:

- l jest etykietowanym hipergrafem lewej strony produkcji,
- r jest etykietowanym hipergrafem prawej strony produkcji,
- $\pi : \hat{\mathscr{H}} \rightarrow \{TRUE, FALSE\}$, jest predykatem określającym (na podstawie danych projektowych) możliwość zastosowania produkcji p,
- $eval$ jest funkcją wyliczającą wartości atrybutów hipergrafu r na podstawie wartości atrybutów podgrafu, do którego dopasowano hipergraf l.

Produkcja p będzie oznaczona $p : l \xrightarrow[eval]{\pi} r$ lub w formie uproszczonej $p : l \longrightarrow r$.

Zastosowanie produkcji $p : l \longrightarrow r$ do zadanego hipergrafu G (np. pokazanego na rysunku 3.13b) składa się z dwóch kroków:

1) wyszukiwany jest podgraf $L \subseteq G$, izomorficzny z l, usuwany z G i zastępowany przez hipergraf r,



Rysunek 3.20. Przykład stacji benzynowej: segmenty definiujące zamknięte przestrzenie ($S_1 \ldots S_9$), lampy ($L_1 \ldots L_{10}$) i sensory (wykrywające obecność: $P_1 \ldots P_6$ lub koniec tankowania: $D_1 \ldots D_8$).

Następnym etapem projektowania jest określenie formalnej reprezentacji rozważanej przestrzeni architektonicznej (zob. rys. 3.20), tj. reprezentacji grafowej. Proponowany model formalny pozwala na wydajne przeprowadzanie obliczeń, szczególnie dla problemów dużej skali [63].

Przykładowa przestrzeń architektoniczna, stacja benzynowa, przedstawiona jest na rysunku 3.20. Wszystkie elementy składowe rozpatrywane są jako bryły (dystrybutory, podpory, zadaszenie, kasy, podłoże) albo bezwymiarowe punkty (punkty świetlne, czujniki). Bryła może być złożona, tj. składać się z elementów będących bryłami prostymi (wielościanami). Reprezentowana jest ona za pomocą hipergrafu (zob. [86]), natomiast bezwymiarowe punkty są dodatkowymi wierzchołkami.

Najistostniejszą cechą modelu hipergrafowego jest umożliwienie wykonywania fotometrycznych obliczeń dwu- i trójwymiarowych. Rezultaty obliczeń mogą być zintegrowane z modelem w postaci atrybutów węzłów lub krawędzi. Zatem są dostępne za jego pośrednictwem.

Rysunek 3.21 przedstawia prostą przestrzeń architektoniczną zawierającą dystrybutor, fragment podłoża oraz zadaszenie wraz z podporami. Wszystkie rozważane obiekty przyjmują formy prostopadłościenne. Dodatkowo istnieją dwa punkty oświetleniowe oraz czujnik. Model grafowy przedstawiony jest na rysunku 3.22. Dla polepszenia czytelności hipergrafy reprezentujące poszczególne bryły są zwinięte i pokazane w formie sześciokątnych węzłów.

Aby osiągnąć cele procesu projektowania (odpowiednie natężenie światła przy minimalnym zużyciu energii), należy rozwiązać problem optymalizacyjny, używając przestrzeni obliczeniowej udostępnianej przez formalny model grafowy.