Reprezentacja wiedzy
Projekt nr 4 — Programy działań z akcjami współbieżnymi
Raport z implementacji i testów

Bartłomiej Dach (szef) Jacek Dziwulski Tymon Felski Jędrzej Fijałkowski
Filip Grajek Maciej Grzeszczak Michał Kołodziej Piotr Piwowarski
Mateusz Rymuszka Piotr Wolski

9 lipca 2018

Niniejszy dokument zawiera raport z implementacji projektu dotyczącego akcji współbieżnych wraz z instrukcją obsługi stworzonego programu. Ponadto w dokumencie zawarty jest raport z fazy testów, podział prac w grupie oraz lista zawartości dołączonej płyty CD.

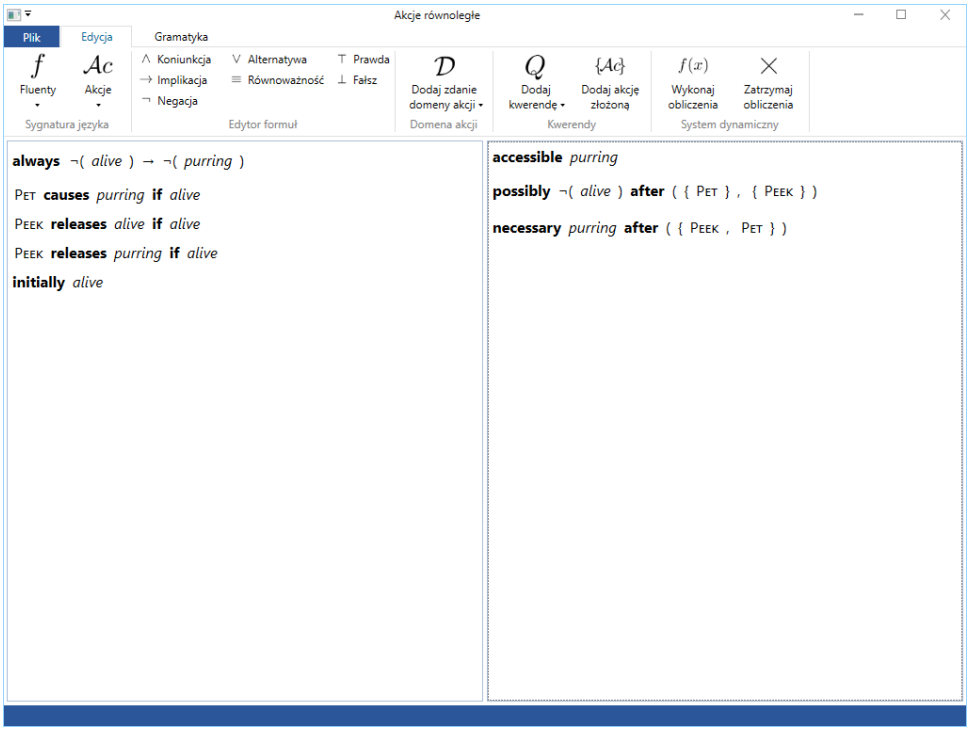
1 Instrukcja obsługi

Do wprowadzania zdań języka akcji oraz kwerend opracowane zostały dwa warianty interfejsu użytkownika:

- Pierwszy wariant, dostępny w zakładce *Edycja*, to typowy edytor graficzny, podobny działaniem do edytora równań w programach pakietu biurowego Microsoft Office. W tym wariancie zdania konstruowane są poprzez zaznaczanie odpowiednich elementów w oknach edycji i użycie opcji na górze okna w celu wypełnienia zdań zgodnie z życzeniem użytkownika.
- Drugą opcją jest zwykły edytor tekstowy, dostępny w zakładce *Gramatyka*. W tym wariancie formuły wprowadzane są w postaci tekstowej i interpretowane z użyciem stworzonej gramatyki bezkontekstowej, która konwertuje zapis zdań na ich postać semantyczną.

Oba rodzaje edytorów opisane są szerzej w poniższych podrozdziałach.

1.1 Edytor graficzny



Rysunek 1: Wygląd edytora graficznego

Edytor graficzny zbliżony jest sposobem działania do wizualnych edytorów równań z innych programów biurowych.

- Wstawianie nowego fluentu do scenariusza wykonywane jest poprzez wciśnięcie przycisku *Fluenty* i wybranie opcji *Dodaj nowy* z menu rozwijanego.

  Analogicznie do scenariusza dodawane są nowe akcje.

- Za wstawianie zdań języka akcji odpowiada przycisk *Dodaj zdanie domeny akcji*. Po wciśnięciu przycisku pojawia się menu rozwijane, z którego należy wybrać żądany rodzaj zdania.

  W analogiczny sposób można dodawać zdania języka kwerend poprzez przycisk *Dodaj kwerendę*.

- Po wybraniu zdania w jednej z list zajmujących główną część okna pojawia się pusta instancja wybranego zdania, z żółtymi znakami zapytania oznaczającymi miejsca, które należy wypełnić.

  – Jeśli w puste miejsce trzeba wstawić fluent, zaznaczamy lukę i wybieramy fluent o żądanej nazwie z menu rozwijanego *Fluenty*.

  – Podobnie, jeśli w puste miejsce trzeba wstawić akcję, zaznaczamy lukę i wybieramy fluent o żądanej nazwie z menu rozwijanego *Akcje*.

  – Jeśli w lukę należy wstawić formułę, po zaznaczeniu luki możemy użyć zarówno fluentów z menu rozwijanego *Fluenty*, jak i przycisków z grupy *Edytor formuł* na górze ekranu.

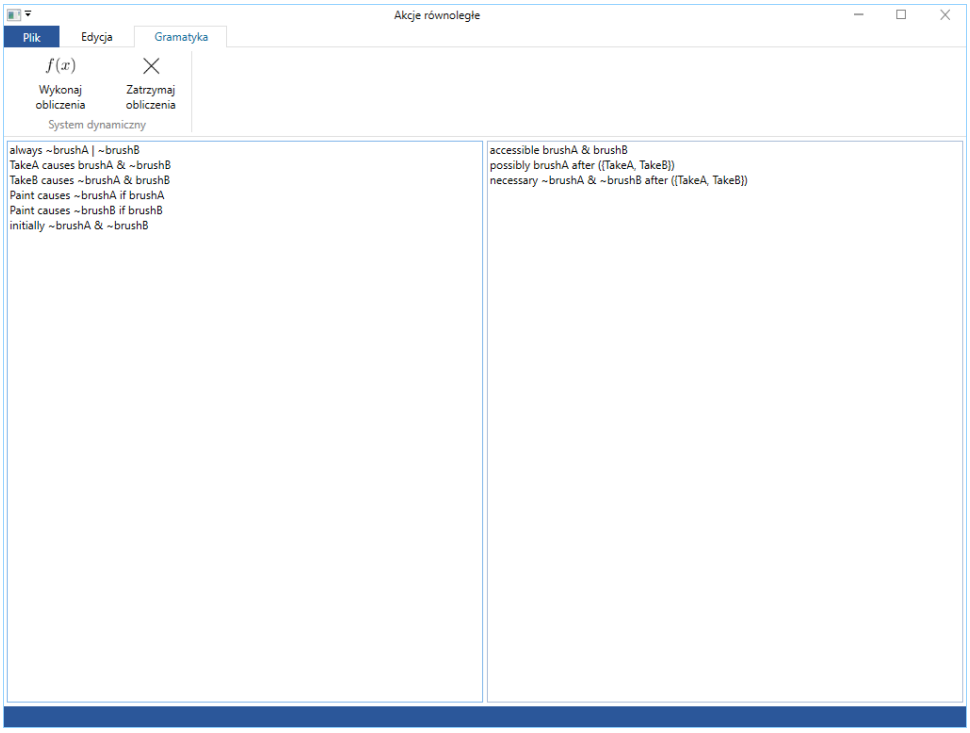
– Aby do kwerendy zawierającej program dodać nową, pustą akcję złożoną, należy zaznaczyć program (oznaczony nawiasami `()`) i wcisnąć przycisk *Dodaj akcję złożoną* żądaną liczbę razy.
– Aby do akcji złożonej dodać akcję atomową, należy zaznaczyć tę akcję (oznaczoną nawiasami `{}`) i wybrać żądaną akcję z menu rozwijanego *Akcje*.

Wstawione akcje, fluenty i formuły można usuwać ze zdań używając przycisku **Delete** na klawiaturze. W ten sam sposób można usuwać całe zdania.

- Aby rozpocząć obliczenia, należy wybrać przycisk *Wykonaj obliczenia*.

Na płycie z programem dołączony został film demonstracyjny, pokazujący sposób obsługi programu.

1.2 Edytor tekstowy



Rysunek 2: Wygląd edytora tekstowego

Edytor tekstowy (rys. 2) stanowi uproszczoną metodę wprowadzania scenariusza do programu. W tym wariancie okno programu składa się z dwóch pól tekstowych:

- Lewe pole tekstowe służy do wprowadzania zdań języka akcji.
- Prawe pole tekstowe służy do wprowadzania zdań języka kwerend.

Składnia wprowadzanych zdań jest zgodna z tą wyspecyfikowaną w dokumentacji teoretycznej. Jedyną różnicą jest wprowadzanie formuł — dla wygody użytkownika przyjęto następujące uproszczenia w notacji logicznej:

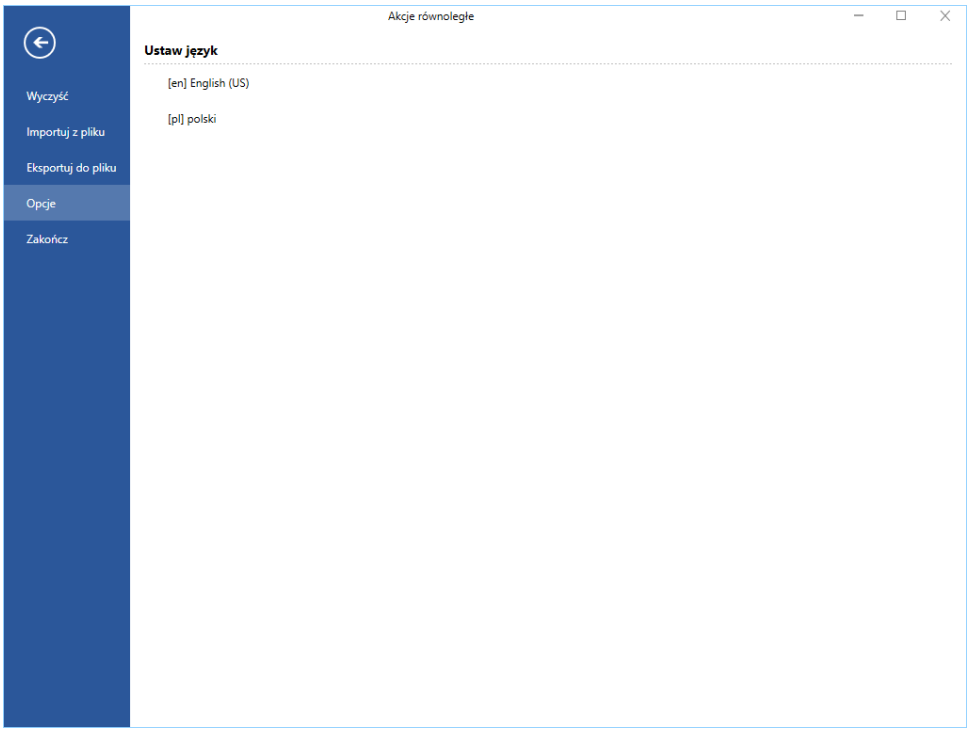
- prawdę ($\top$) zapisujemy `T` (**T**ruth),
- fałsz ($\bot$) zapisujemy `F` (**F**alsity),
- negację ($\neg p$) zapisujemy używając znaku tyldy: `~p`,
- koniunkcję ($p \wedge q$) zapisujemy `p & q`,
- alternatywę ($p \vee q$) zapisujemy `p | q`,
- implikację ($p \to q$) zapisujemy `p -> q`,
- równoważność ($p \equiv q$) zapisujemy `p <-> q`.

Ewentualne błędy wykryte przez program w procesie analizy składniowej wprowadzonych zdań wyświetlane są na pasku statusu programu, na dole okna.

Do programu zostały dołączone przykładowe pliki tekstowe demonstrujące format wejściowy zdań dla edytora tekstowego.

Tak samo, jak w przypadku edytora graficznego, za obliczenie wartości logicznej kwerend odpowiada przycisk *Wykonaj obliczenia* w górnej części okna. Długotrwałe obliczenia można przerwać przyciskiem *Zatrzymaj obliczenia*.

1.3 Dostępne opcje



Rysunek 3: Menu opcji programu

Wybranie zakładki *Plik* w górnym menu spowoduje przejście do menu opcji programu, widocznego na rysunku 3. Poniżej znajduje się opis dostępnych opcji.

- Opcja *Wyczyść* powoduje usunięcie wszystkich fluentów, akcji i zdań z obecnego scenariusza.
- Opcja *Importuj z pliku* umożliwia wczytanie wcześniej zapisanego scenariusza z pliku w formacie `.xml` lub `.txt`. Przykładowe pliki wejściowe dla programu zostały umieszczone na załączonej płycie CD.
- Opcja *Eksportuj do pliku* umożliwia zapis obecnie edytowanego scenariusza do pliku. Scenariusze edytowane w trybie graficznym eksportowane są do formatu `.xml`, zaś scenariusze edytowane w trybie tekstowym — do formatu `.txt`.
- W podmenu *Opcje* można zmienić język stosowany w interfejsie. Dostępne opcje to: język angielski i polski.
- Opcja *Zakończ* powoduje wyjście z programu.

2 Raport z przeprowadzonych testów

Zadaniem zespołu było przetestowanie aplikacji opracowanej przez grupę nr 3. Tematem projektu nr 3 były **programy działań w środowisku wieloagentowym**.

Zespół nr 3 przekazał wersję aplikacji do testów 4 czerwca. W trakcie analizy dokumentacji i programu wyszczególnione zostały następujące uwagi:

- **Edytor czasem nie wprowadza zmian w formułach i kwerendach**
  Po użyciu przycisku do edycji zdań w scenariuszu pojawia się okno dialogowe pozwalające na zmianę zawartości zdania. Po wprowadzeniu żądanych zmian i wciśnięciu *Ok* zmiany nie pojawiają się w głównym oknie aplikacji. Przy ponownej próbie edycji tego samego zdania wprowadzone zmiany są jednak widoczne.

  Wciśnięcie przycisku odświeżania po prawej stronie opcji *Result* w oknie dialogowym powoduje, że wprowadzone zmiany są widoczne wszędzie, lecz naszym zdaniem nie jest intuicyjne.
- **Awaria aplikacji po usunięciu wszystkich fluentów i agentów**
  Po usunięciu wszystkich fluentów i agentów ze scenariusza i wciśnięciu przycisku *Compute* program niespodziewanie się wyłącza.
- **Nazwy fluentów, akcji i agentów nie są unikalne, powodują awarię aplikacji**
  Możliwe jest dodanie fluentu, akcji i agenta o tej samej nazwie, co jeden z istniejących. Po dodaniu duplikatu i wciśnięciu przycisku *Compute* program niespodziewanie się wyłącza.
- **Błędny wynik zapytania w historyjce 1. w porównaniu z wynikiem w dokumentacji**
  W historyjce 1, w dokumentacji zapytanie:

  $$\textbf{necessary accessible } (M_l) \textbf{ from } (\neg M_l) \textbf{ by } M$$

  ma wynik FALSE, podczas gdy w programie zwrócone jest TRUE, mimo że formuły w dokumentacji i programie są takie same dla historyjki 1.

- **Awaria aplikacji po zmianie typu istniejącej kwerendy**
  Przy edycji dowolnej kwerendy, zmiana jej typu, wciśnięcie *Ok*, a następnie *Compute* powoduje nieoczekiwany błąd programu.

- **Błędne wyniki kwerend po modyfikacji historii 1.**
  Jeśli po załadowaniu w programie historii 1. usuniemy zdanie

  $$\textsc{Attack} \textbf{ by } (M) \textbf{ releases } S_i \textbf{ if } (M_l, S_i)$$

  zmienimy formułę **initially** S_i na **initially** $\neg S_i$ i wciśniemy *Compute*, to zapytanie

  $$\textbf{possibly } \neg S_i \textbf{ after } (\textsc{Attack} \textbf{ by } (M)) \textbf{ from } (M_i)$$

  zwróci wynik False, chociaż oczekiwany jest wynik True, ponieważ S_i ma wartość początkową False. Natomiast wstawienie zapytania

  $$\textbf{necessary } \neg S_i \textbf{ after } (\textsc{Attack} \textbf{ by } (M)) \textbf{ from } (M_i)$$

  powoduje nieoczekiwane zakończenie programu.

- **Sprzeczne definicje prawdziwości zdania impossible**
  Na stronie 5. dokumentacji dana jest definicja prawdziwości zdania **impossible**:

  > Wyrażenie (**impossible** A **by** G **if** π) jest prawdziwe w S wtedy i tylko wtedy, gdy
  > $$\forall \sigma \models \pi \rightarrow Res(A, G, \sigma) = \{\sigma\}$$

  Natomiast ponieważ zdanie

  $$\textbf{impossible } A \textbf{ by } G \textbf{ if } \pi$$

  jest alternatywnym zapisem zdania

  $$A \textbf{ by } G \textbf{ causes } \bot \textbf{ if } \pi$$

  to z definicji funkcji Res_0 wynika, że dla akcji A objętej zdaniem **impossible**, pod warunkiem spełnienia warunku wstępnego π, dla dowolnego stanu σ' następnik implikacji w definicji Res_0 jest fałszywy, zatem

  $$Res_0(A, G, \sigma') = Res(A, G, \sigma') = \emptyset$$

- **Historia 2. — wynik sprzeczny z grafem przejść w dokumentacji**
  Dla przykładu 2. (sekcja 4. w dokumentacji) graf funkcji przejścia przedstawiony w podsekcji 4.7. pokazuje, że ze stanu (L, P) nie powinno być możliwe osiągnięcie stanu $(\neg L, P)$ Tymczasem wprowadzenie do programu kwerend postaci

  $$\textbf{possibly accessible } (\neg L, P) \textbf{ from } (L, P) \textbf{ by } \{J\}$$

  $$\textbf{necessary accessible } (\neg L, P) \textbf{ from } (L, P) \textbf{ by } \{J\}$$

  powoduje uzyskanie dwóch wyników True.

- **Edytor scenariusza nie pozwala na wprowadzenie dowolnej formuły**
  Dokumentacja specyfikuje, że w zdaniach wartości, obserwacji, ograniczeń i efektu można wstawiać formuły logiczne. Edytor zdań scenariusza pozwala jednak tylko na wstawianie literałów.

- **Brak opisu fluentu, akcji lub agenta powoduje awarię aplikacji**
  Usunięcie wszystkich agentów, formuł i akcji, dodanie nowych bez opisu, a następnie naciśnięcie przycisku *Compute* program niespodziewanie się wyłącza.

3 Zgłoszone błędy w aplikacji

Stworzona aplikacja dotycząca akcji współbieżnych została przekazana do testów zespołowi nr 5 26 maja. Poniżej znajduje się podsumowanie zgłoszonych błędów wraz z opisem sposobu ich rozwiązania.

- **Dodanie niezależnego nieinercjalnego fluentu zmienia wynik kwerendy**
  - **Opis:** Dodanie zdania ***noninertial*** c do domeny akcji

    $$\begin{aligned} &\textsc{doA}\ \textbf{\textit{causes}}\ a\ \textbf{\textit{if}}\ b \\ &\textbf{\textit{initially}}\ \neg a \\ &\textsc{doB}\ \textbf{\textit{releases}}\ b \end{aligned}$$

    zmienia wynik kwerendy ***accessible*** a z True na False.
  - **Sposób rozwiązania:** Podczas analizy błędu okazało się, że sposób ewaluacji kwerendy ***accessible*** był niezgodny z opisem teoretycznym, ponieważ dla danego stanu sprawdzane było, czy ze wszystkich możliwych stanów wynikowych można osiągnąć cel γ (zamiast czy można ten cel osiągnąć z pewnego jednego stanu wynikowego). Błąd został naprawiony.
- **Długotrwałe obliczenia**
  - **Opis:** Dla większych sygnatur języka (złożonych z 5 akcji i 3 fluentów) ewaluacja kwerendy typu

    $$\textbf{\textit{accessible}}\ \bot$$

    trwa długo (ponad 5 minut)
  - **Sposób rozwiązania:** Zdaniem zespołu przyczyną jest wykładnicza złożoność znajdowania odpowiedzi na kwerendy. Dla 5 akcji atomowych i 3 fluentów otrzymujemy $2^5 = 32$ akcje złożone i $2^3 = 8$ fluentów. Ponieważ kwerendy typu ***accessible*** $\bot$ wymagają przejrzenia wszystkich możliwych kombinacji stanów i akcji, ewaluacja kwerendy sprowadza się do rekurencji o głębokości 8 i stopniu rozgałęzienia 32. Uznajemy, że program działa prawidłowo. Do aplikacji została jednak dodana opcja anulowania trwających obliczeń w przypadku długotrwałego oczekiwania na wynik.

Ponadto, podczas checkpointu 4. zaobserwowane zostały następujące wyniki programu:

- **Nieprawidłowy sposób uwzględniania dekompozycji akcji**
  - **Opis:** Gdy dana jest dziedzina akcji

    $$\begin{aligned} &\textbf{\textit{initially}}\ hasKey \\ &\textsc{Open}\ \textbf{\textit{causes}}\ doorOpen\ \textbf{\textit{if}}\ hasKey \\ &\textsc{Close}\ \textbf{\textit{causes}}\ \neg doorOpen\ \textbf{\textit{if}}\ hasKey \end{aligned}$$

    kwerendy

    $$\begin{aligned} &\textbf{\textit{possibly}}\ doorOpen\ \textbf{\textit{after}}(\{\textsc{Open}, \textsc{Close}\}) \\ &\textbf{\textit{possibly}}\ \neg doorOpen\ \textbf{\textit{after}}(\{\textsc{Open}, \textsc{Close}\}) \end{aligned}$$

    zwracają wynik False.

– **Sposób rozwiązania:** Problemem był niewłaściwy sposób uwzględniania dekompozycji akcji. Dokładniej, dekompozycje były wyznaczane i uwzględniane na etapie obliczania funkcji Res_0, co oznacza, że dekompozycje podlegały minimalizacji zmian. Powodowało to, że w zależności od stanu początkowego w modelu, wynik jednej z akcji OPEN lub CLOSE był ignorowany.
Problem został rozwiązany.

- **Nieintuicyjny wynik zapytania w historii z alternatywą**
  – **Opis:** Gdy dana jest dziedzina akcji

$$\begin{aligned} &\textbf{\textit{initially}}\ f \\ &A\ \textbf{\textit{causes}}\ g \vee h\ \textbf{\textit{if}}\ f \\ &B\ \textbf{\textit{causes}}\ g \vee h\ \textbf{\textit{if}}\ f \end{aligned}$$

kwerendy

$$\begin{aligned} &\textbf{\textit{possibly}}\ g\ \textbf{\textit{after}}(\{A, B\}) \\ &\textbf{\textit{possibly}}\ h\ \textbf{\textit{after}}(\{A, B\}) \end{aligned}$$

zwracają wynik FALSE.
– **Sposób rozwiązania:** W tym przypadku program działa poprawnie. Przyczyną wyniku FALSE jest minimalizacja zmian.
A, B są akcjami konfliktowymi, więc wykonanie $\{A, B\}$ wiąże się z niedeterministycznym wykonaniem A lub B. Ponieważ fluenty g, h są inercjalne, podlegają minimalizacji zmian. Załóżmy przykładowo, że dany jest stan początkowy $\sigma_0 = (f, \neg g, h)$. Mamy wówczas

$$\begin{aligned} Res_0((f, \neg g, h), \{A\}) &= \{(f, \neg g, h), (f, g, \neg h), (f, g, h)\} \\ Res_0((f, \neg g, h), \{B\}) &= \{(f, \neg g, h), (f, g, \neg h), (f, g, h)\} \end{aligned}$$

$$\begin{aligned} (*) \quad New(A, (f, \neg g, h), (f, \neg g, h)) &= \emptyset \\ New(A, (f, \neg g, h), (f, g, \neg h)) &= \{g, \neg h\} \\ New(A, (f, \neg g, h), (f, g, h)) &= \{h\} \end{aligned}$$

$$\begin{aligned} Res_0((f, \neg g, h), \{A\}) &= \{(f, \neg g, h)\} \\ Res_0((f, \neg g, h), \{B\}) &= \{(f, \neg g, h)\} \end{aligned}$$

Wobec tego istnieje model D taki, w którym zachodzi $\neg g$ po wykonaniu $\{A, B\}$. Kwerenda $\textbf{\textit{possibly}}\ g\ \textbf{\textit{after}}(\{A, B\})$ jest więc fałszywa. W analogiczny sposób fałszywa jest również kwerenda $\textbf{\textit{possibly}}\ h\ \textbf{\textit{after}}(\{A, B\})$. Kwerendy

$$\begin{aligned} &\textbf{\textit{possibly}}\ g \vee h\ \textbf{\textit{after}}(\{A, B\}) \\ &\textbf{\textit{necessary}}\ g \vee h\ \textbf{\textit{after}}(\{A, B\}) \end{aligned}$$

są jednak prawdziwe.
Dodanie do domeny akcji zdań

$$\begin{aligned} &\textbf{\textit{noninertial}}\ g \\ &\textbf{\textit{noninertial}}\ h \end{aligned}$$

powoduje uzyskanie intuicyjnego wyniku TRUE na dane kwerendy.

4 Podział prac

W implementacji aplikacji brali udział wszyscy członkowie zespołu. Poniżej znajduje się szczegółowy opis podziału prac w fazie implementacyjnej.

- **Bartłomiej Dach** przygotował model danych aplikacji.
- **Tymon Felski** i **Bartłomiej Dach** przygotowali interfejs użytkownika.
- **Tymon Felski** i **Piotr Piwowarski** opracowali zapisywanie i wczytywanie scenariuszy z pliku.
- **Jacek Dziwulski** przygotował komponent generujący zbiór wszystkich możliwych stanów oraz zbiór wszystkich akcji złożonych.
- **Maciej Grzeszczak** przygotował część systemu sprowadzającą formuły logiczne do dysjunkcyjnej postaci normalnej, opracował gramatykę bezkontekstową dla języka akcji i kwerend w postaci rozszerzonej notacji Backusa-Naura (ang. EBNF — *Extended Backus-Naur Form*) i zintegrował ją z interfejsem użytkownika.
- **Filip Grajek** przygotował komponent obliczający dekompozycje poszczególnych akcji złożonych.
- **Piotr Wolski** przygotował komponent wyznaczający wartości funkcji Res_0.
- **Jędrzej Fijałkowski** przygotował komponent wyznaczający zbiory New.
- **Michał Kołodziej** i **Mateusz Rymuszka** przygotowali część systemu obliczającą wartości funkcji Res i ewaluującą kwerendy.
- **Piotr Piwowarski** przygotował część systemu sprawdzającą prawdziwość zdań wartości i obserwacji w domenie akcji.

Ponadto **Bartłomiej Dach**, **Piotr Wolski**, **Michał Kołodziej** i **Jacek Dziwulski** brali udział w testach aplikacji zespołu nr 3, których wynikiem były uwagi zawarte w rozdziale 2.

Bartłomiej Dach, **Filip Grajek** i **Jędrzej Fijałkowski** brali udział przy analizie i naprawie błędów znalezionych przez zespół nr 5 oraz wykrytych podczas checkpointu 4.

5 Opis zawartości płyty CD

Płyta CD dołączona do raportu zawiera następujące foldery:

- Folder `bin/` zawiera gotową wykonywalną wersję aplikacji. Aby uruchomić program, należy wybrać plik `bin/ConcurrentActions.exe`.
- Folder `demo/` zawiera demonstracyjny materiał filmowy pokazujący sposób użytkowania stworzonego programu.
- Folder `doc/` zawiera całość dokumentacji projektowej, tj. podstawy teoretyczne projektu oraz niniejszy raport z implementacji.

- W folderze `input/` znajdują się przykładowe pliki wejściowe dla programu, w formatach `.xml` oraz `.txt`.
- Katalog `src/` zawiera kod źródłowy opracowanej aplikacji.