Reprezentacja wiedzy
Projekt nr 4. — Programy działań z akcjami współbieżnymi
Podstawy teoretyczne

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

9 lipca 2018

1 Założenia

Dana jest klasa systemów dynamicznych spełniających następujące warunki:

1. Prawo inercji.
2. Niedeterminizm.
3. W języku kwerend występują akcje złożone (zbiory co najwyżej k akcji atomowych), w języku akcji jedynie akcje atomowe.
4. Pełna informacja o wszystkich akcjach atomowych i wszystkich ich skutkach bezpośrednich.
5. Z każdą akcją atomową związany jest jej warunek wstępny (ew. TRUE) i końcowy (efekt akcji).
6. Wykonywane są jedynie akcje bezkonfliktowe (żadne dwie akcje składowe nie mogą mieć wspólnych zmiennych, na które w żadnym stanie mają wpływ).
7. Wynikiem akcji złożonej jest suma skutków wszystkich składowych akcji bezkonfliktowych.
8. Akcje mogą być niewykonalne w pewnych stanach; jeśli akcja jest niewykonywalna, to każda akcja ją zawierająca też jest niewykonalna.
9. Dopuszczalny jest opis częściowy zarówno stanu początkowego, jak i pewnych stanów wynikających z wykonań sekwencji akcji.

Definicja 1. ***Programem*** P ***działań*** *jest ciąg* $P = (A_1, \ldots, A_n)$ *akcji złożonych.*

Definicja 2. *Program* $P = (A_1, \ldots, A_n)$ *jest* ***realizowalny****, jeśli wszystkie akcje złożone* A_i *są wykonalne.*

Celem projektu jest opracowanie i zaimplementowanie języka akcji dla specyfikacji podanej klasy systemów dynamicznych oraz odpowiadającego mu języka kwerend zapewniającego uzyskanie odpowiedzi na następujące zapytania:

1. Czy podany program P działań jest możliwy do realizacji zawsze/kiedykolwiek ze stanu początkowego?
2. Czy wykonanie programu P w stanie początkowym działań prowadzi zawsze/kiedykolwiek do osiągnięcia celu γ?
3. Czy cel γ jest osiągalny ze stanu początkowego?

2 Język akcji

Scenariusze dla zadanej klasy systemów dynamicznych będą reprezentowane za pomocą języka akcji wzorowanego na językach z rodziny $\mathcal{AR}$. Zgodnie z założeniami projektu, w języku akcji będą rozważane jedynie następstwa akcji atomowych.

2.1 Składnia języka

Definicja 3. ***Sygnaturą*** *języka jest uporządkowana para zbiorów $\Upsilon = (\mathcal{F}, \mathcal{A}c)$, gdzie $\mathcal{F}$ to zbiór fluentów, a $\mathcal{A}c$ to zbiór akcji. Dodatkowo definiuje się zbiór* ***fluentów inercjalnych*** *$\mathcal{F}_I \subseteq \mathcal{F}$.*

Sygnatura języka określa zarówno logiczne zmienne określające stan świata (fluenty), jak i możliwe metody manipulacji tym światem (akcje).

Definicja 4. ***Literałem*** *$\overline{f}$ będziemy nazywać jedno z wyrażeń*

$$\overline{f} ::= f \mid \neg f$$

gdzie $f \in \mathcal{F}$.

Definicja 5. ***Formuła*** *nad zbiorem fluentów $\mathcal{F}$ to dowolny predykat zbudowany z fluentów:*

$$\alpha ::= f \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \rightarrow \alpha_2 \mid \alpha_1 \equiv \alpha_2$$

Ponadto definiujemy dwie specjalne formuły:

- $\top$: *prawda,*
- $\bot$: *fałsz.*

Zbiór wszystkich formuł nad zbiorem fluentów $\mathcal{F}$ oznaczamy jako $Forms(\mathcal{F})$.

Proponowany język definiuje następujące siedem typów wyrażeń:

Definicja 6. ***Zdaniem wartości początkowej*** *nazywamy zdanie postaci*

$$\textbf{\textit{initially}}\,\alpha$$

gdzie $\alpha \in Forms(\mathcal{F})$.

Zdanie wartości początkowej będzie służyło do określenia początkowego stanu fluentów. Zgodnie z założeniami, w systemie nie jest wymagane wyspecyfikowanie wartości początkowej wszystkich fluentów.

Definicja 7. ***Zdaniem wartości*** *nazywamy zdanie postaci*

$$\alpha \,\textbf{\textit{after}}\, A$$

gdzie $\alpha \in Forms(\mathcal{F})$ *oraz* $A \in \mathcal{A}c$.

Definicja 8. ***Zdaniem obserwacji*** *nazywamy zdanie postaci*

$$\textbf{\textit{observable}}\, \alpha \,\textbf{\textit{after}}\, A$$

gdzie $\alpha \in Forms(\mathcal{F})$ *oraz* $A \in \mathcal{A}c$.

Zdanie wartości mówi, że warunek α musi zachodzić po wykonaniu akcji A, z kolei zdanie obserwacji określa, że warunek ten może być spełniony, ale niekoniecznie musi.

Definicja 9. ***Zdaniem efektu*** *nazywamy zdanie postaci*

$$A \,\textbf{\textit{causes}}\, \alpha \,\textbf{\textit{if}}\, \pi$$

gdzie $\alpha, \pi \in Forms(\mathcal{F})$ *oraz* $A \in \mathcal{A}c$.

W szczególności:

- *jeśli* $\pi = \top$, *to piszemy, że* $A \,\textbf{\textit{causes}}\, \alpha$,
- *jeśli* $\alpha = \bot$, *to piszemy, że* $\textbf{\textit{impossible}}\, A \,\textbf{\textit{if}}\, \pi$.

Zdania efektu określają następstwa akcji, gdy spełniony jest warunek α, zwany również **warunkiem wstępnym** akcji. Szczególną wagę ma zdanie ***impossible***, które może uniemożliwić wykonanie danej akcji atomowej pod warunkiem spełnienia warunku wstępnego.

Definicja 10. ***Zdaniem uwolnienia fluentu*** *nazywamy zdanie postaci*

$$A \,\textbf{\textit{releases}}\, f \,\textbf{\textit{if}}\, \pi$$

gdzie $\pi \in Forms(\mathcal{F})$, $f \in \mathcal{F}_I$ *oraz* $A \in \mathcal{A}c$.

W szczególności, jeśli $\pi = \top$, *to piszemy* $A \,\textbf{\textit{releases}}\, f$.

Zdanie to należy interpretować tak, że wykonanie akcji A w każdym stanie może, ale nie musi zmienić wartości fluentu inercjalnego f. Innymi słowy, może ono służyć do modelowania akcji losowych.

Definicja 11. ***Zdaniem ograniczenia*** *nazywamy zdanie postaci*

$$\textbf{\textit{always}}\, \alpha$$

gdzie $\alpha \in Forms(\mathcal{F})$.

Ograniczenia stosowane są do wyeliminowania pewnych niepożądanych stanów systemu. Wyrażenie α zwane jest w szczególności **warunkiem integralności**.

Definicja 12. ***Zdaniem specyfikacji fluentu*** *nazywamy zdanie postaci*

$$\textbf{\textit{noninertial}}\, f$$

gdzie $f \notin \mathcal{F}_I$.

Powyższe zdanie można traktować jak deklarację istnienia nieinercjalnego fluentu w danej sygnaturze.

Definicja 13. *Domeną akcji* $\mathcal{D}$ *nazywamy skończony zbiór zdań zdefiniowanych powyżej.*

2.2 Semantyka języka

Definicja 14. *Stanem nazywamy funkcję* $\sigma : \mathcal{F} \to \{0,1\}$.

Dodatkowo, przez $\sigma^* : Forms(\mathcal{F}) \to \{0,1\}$ *określamy rozszerzenie tej funkcji, zdefiniowane dla wszystkich formuł nad zbiorem fluentów: gdzie*

$$\begin{aligned}
\sigma^*(f) &= \sigma(f) \ \ dla\ f \in \mathcal{F} \\
\sigma^*(\neg\alpha) &= 1 - \sigma^*(\alpha) \\
\sigma^*(\alpha \wedge \beta) &= \min(\sigma^*(\alpha), \sigma^*(\beta)) \\
\sigma^*(\alpha \vee \beta) &= \max(\sigma^*(\alpha), \sigma^*(\beta)) \\
\sigma^*(\alpha \to \beta) &= \sigma^*(\neg\alpha \vee \beta) \\
\sigma^*(\alpha \equiv \beta) &= \sigma^*((\alpha \to \beta) \wedge (\beta \to \alpha))
\end{aligned}$$

Dla każdej formuły $\alpha \in Forms(\mathcal{F})$, *gdy* $\sigma^*(\alpha) = 1$, *piszemy* $\sigma \models \alpha$.

Funkcja stanu definiuje w sposób jednoznaczny wartość wszystkich fluentów.

Definicja 15. ***Strukturą*** *dla języka* $\mathcal{L}$ *nazywamy trójkę*

$$\mathcal{S} = (\Sigma, \sigma_0, Res)$$

gdzie:

1. Σ *to zbiór stanów,*
2. $\sigma_0 \in \Sigma$ *to stan początkowy,*
3. $Res : 2^{\mathcal{A}c} \times \Sigma \to 2^{\Sigma}$ *to funkcja przejścia.*

Docelowo, dla każdej akcji złożonej $A' \in 2^{\mathcal{A}c}$ i dla każdego stanu $\sigma \in \Sigma$, Res przyporządkuje zbiór stanów możliwych do osiągnięcia przez wykonanie akcji A' w stanie σ.

Zauważmy, że docelowo w systemie może być rozważane wiele struktur dla jednej domeny akcji, różniących się stanem początkowym, z racji niepełnej wiedzy o początkowej wartości fluentów.

Funkcję przejścia możemy rozszerzyć na pojęcie programu w następujący sposób:

Definicja 16. *Niech* $\mathcal{S} = (\Sigma, \sigma_0, Res)$ *będzie strukturą. Niech* $Res^* : (2^{\mathcal{A}c})^* \times \Sigma \to 2^{\Sigma}$ *będzie funkcją taką, że:*

- $Res^*(\varepsilon, \sigma) = \{\sigma\}$,
- $Res^*((A_1), \sigma) = Res(A_1, \sigma)$,
- $Res^*((A_1, A_2, \ldots, A_n, A_{n+1}), \sigma) = \bigcup_{\sigma' \in Res^*((A_1, A_2, \ldots, A_n), \sigma)} Res(A_{n+1}, \sigma')$.

Wtedy pojedynczą ścieżkę wykonania programu możemy odtworzyć za pomocą funkcji $\Psi_{\mathcal{S}}$ zdefiniowaną następująco:

Definicja 17. *Niech* $\mathcal{D}$ *będzie domeną akcji. Dla struktury* $S = (\Sigma, \sigma_0, Res)$, *definiujemy funkcję częściową* $\Psi_{\mathcal{S}} : (2^{\mathcal{A}c})^* \times \Sigma \to \Sigma$ *w następujący sposób:*

- $\Psi_{\mathcal{S}}(\varepsilon, \sigma) = \sigma$
- *jeśli dla* $n \geq 1, \Psi_{\mathcal{S}}((A_1, \ldots, A_n), \sigma)$ *jest zdefiniowane i* $Res^*((A_1, \ldots, A_n, A_{n+1}), \sigma) \neq \emptyset$, *to* $\Psi_{\mathcal{S}}((A_1, \ldots, A_n, A_{n+1}), \sigma) \in Res(A_{n+1}, \Psi_{\mathcal{S}}((A_1, \ldots, A_n), \sigma))$.

2.2.1 Akcje złożone

Zanim zdefiniujemy model dla domen akcji rozważanego języka, należy określić, na czym polega wykonanie akcji złożonej z wielu akcji atomowych. Zauważmy, że w domenie akcji mogą znajdować się zdania, których efekty będą ze sobą wzajemnie sprzeczne, tj. dla pewnego zbioru akcji atomowych i pewnego stanu nie będzie możliwe spełnienie ich warunków końcowych. Do sformalizowania tej intuicji posłużymy się dysjunkcyjną postacią normalną formuł rachunku zdań.

Definicja 18. *Niech α będzie dowolną formułą nad zbiorem fluentów $\mathcal{F}$. Mówimy, że α jest w **dysjunkcyjnej postaci normalnej (DNF)** wtedy i tylko wtedy, gdy*

$$\alpha = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{k(i)} \lambda_{ij}$$

gdzie λ_{ij} jest literałem.

Twierdzenie 1. *Każdą formułę α można przekształcić do formuły α', która jest w postaci DNF.*

Formułę α przekształconą do postaci DNF oznaczamy jako $DNF(\alpha)$.

Definicja 19. *Niech $DNF(\alpha) = \gamma_1 \vee \cdots \vee \gamma_k$. Każdą formułę γ_i nazywamy **komponentem** formuły α.*

Po zdefiniowaniu DNF przystąpmy do określenia, jakie akcje będą rozważane za konfliktowe.

Definicja 20. *Niech $\mathcal{D}$ będzie domeną akcji, σ będzie pewnym stanem oraz niech $A, B \in \mathcal{A}c$ będą akcjami elementarnymi. Akcje A, B są **konfliktowe** wtedy i tylko wtedy, gdy przynajmniej jeden z następujących warunków jest prawdziwy:*

- *W D znajdują się dwa zdania postaci:*

  $$A \textbf{ causes } \alpha \textbf{ if } \pi_1$$
  $$B \textbf{ causes } \beta \textbf{ if } \pi_2$$

  takie, że:

  - $\sigma \models \pi_1 \wedge \pi_2$
  - *Jakiś komponent z α jest sprzeczny z jakimś komponentem z β.*

- *W D znajdują się dwa zdania postaci:*

  $$A \textbf{ causes } \alpha \textbf{ if } \pi_1$$
  $$B \textbf{ releases } f \textbf{ if } \pi_2$$

  takie, że:

  - $\sigma \models \pi_1 \wedge \pi_2$
  - *w jednym z komponentów α występuje f.*

- *W D znajdują się dwa zdania postaci:*

  $$A \textbf{ releases } f \textbf{ if } \pi_1$$
  $$B \textbf{ releases } f \textbf{ if } \pi_2$$

  takie, że $\sigma \models \pi_1 \wedge \pi_2$.

Zauważmy, że w pierwszym przypadku warunek o sprzeczności komponentów z α i β nie jest równoważny sprzeczności formuły $\alpha \wedge \beta$.

Definicja 21. *Niech $\mathcal{D}$ to domena akcji, niech σ to pewien stan. Akcja złożona $\{A_1, \ldots A_k\}$, gdzie A_i to akcje elementarne, jest* ***potencjalnie wykonywalna*** *w σ wtedy i tylko wtedy gdy nie istnieje $B \subseteq A_1 \cup \cdots \cup A_k$ takie, że $\mathcal{D}$ zawiera zdania:*

$$B \textbf{ causes } \alpha_1 \textbf{ if } \pi_1$$
$$\ldots$$
$$B \textbf{ causes } \alpha_k \textbf{ if } \pi_k$$

takie, że:

- $\sigma \models \pi_1 \wedge \cdots \wedge \pi_k$
- $\alpha_1 \wedge \cdots \wedge \alpha_k$ *jest sprzeczne.*

Definicja 22. *Niech $\mathcal{D}$ będzie domeną akcji, niech σ będzie stanem oraz niech $A \subseteq \mathcal{A}c$.* ***Dekompozycją*** *A w σ będziemy nazywać taki zbiór $\Delta(A, \sigma) = \{E_1, E_2, \ldots, E_k\}$ akcji elementarnych w D, że:*

1. $\bigcup_{i=1}^{k} E_i \subseteq A$,
2. $\Delta(A, \sigma)$ *jest zbiorem niekonfliktujących akcji potencjalnie wykonywalnych w σ,*
3. *Nie ma żadnego zbioru $\{E'_1, \ldots, E'_k, E'_{k+1}, \ldots, E'_m\}$ akcji spełniającego powyższe warunki takiego, że $E_i \subset E'_i$ dla pewnego $i = 1, \ldots, k$.*

Innymi słowy, dekompozycja akcji złożonej A określa maksymalne ze względu na inkluzję potencjalnie wykonywane podakcje A. Zauważmy, że na mocy definicji 20 i 21 możemy wspomóc się dysjunkcyjną postacią normalną przy wyznaczaniu niekonfliktujących akcji.

Definicja 23. *Niech $\Delta(A, \sigma) = \{E_1, E_2, \ldots, E_k\}$ będzie dekompozycją A w σ. Każdą akcję E_i nazywamy* ***komponentem wykonania*** *A w stanie σ.*

Definicja 24. *Niech $\mathcal{D}$ będzie domeną akcji, niech σ, σ' będą stanami, niech $A \subseteq \mathcal{A}c$ będzie akcją i niech $\Delta(A, \sigma) = \{E_1, E_2, \ldots, E_k\}$ będzie dekompozycją A w σ. Mówimy, że σ' jest* ***potencjalnym wynikiem wykonania*** *A w σ wtedy i tylko wtedy gdy dla każdego $i = 1, \ldots, k$ i dla każdego zdania postaci:*

$$(E_i \textbf{ causes } \alpha \textbf{ if } \pi) \in D$$

zachodzi $\sigma \models \pi \Rightarrow \sigma' \models \alpha$.

Oznacza to, że po wykonaniu A w stanie σ stan wynikowy to taki, który spełnia wszystkie efekty każdego komponentu E_i z danej dekompozycji. Wybór dekompozycji jest jednym z elementów niedeterminizmu w rozważanym systemie.

Zauważmy ponadto, że takie zdefiniowanie potencjalnego wyniku wykonania akcji zapewnia, że jeśli akcja złożona zawiera akcję atomową niewykonalną w σ, to cała akcja złożona również nie będzie wykonalna.

2.2.2 Model domeny akcji

Niech $\mathcal{D}$ będzie domeną akcji, zaś $\mathcal{S} = (\Sigma, \sigma_0, Res)$ pewną strukturą języka.

Definicja 25. *Fluent $f \in \mathcal{F}$ jest inercjalny w $\mathcal{S}$ wtedy i tylko wtedy, gdy $(\textbf{\textit{noninertial}}\, f) \notin \mathcal{D}$.*

Definicja 26. *$\sigma : \mathcal{F} \to \{0, 1\}$ jest **stanem** $\mathcal{S}$ względem $\mathcal{D}$ wtedy i tylko wtedy, gdy dla każdego zdania ograniczenia $(\textbf{\textit{always}}\, \alpha) \in \mathcal{D}$ zachodzi $\sigma \models \alpha$.*

Powyższa definicja stanu względem $\mathcal{D}$ zapewni spełnienie warunków integralności w każdym modelu domeny $\mathcal{D}$.

Definicja 27. *Zdanie wartości $(\alpha\, \textbf{\textit{after}}\, A)$ jest **prawdziwe** w $\mathcal{S}$ wtedy i tylko wtedy, gdy*

$$\Psi_{\mathcal{S}}(\{A\}, \sigma_0) \models \alpha$$

dla każdej funkcji $\Psi_{\mathcal{S}}$.

Definicja 28. *Zdanie obserwacji $(\textbf{\textit{observable}}\, \alpha\, \textbf{\textit{after}}\, A)$ jest **prawdziwe** w $\mathcal{D}$ wtedy i tylko wtedy, gdy*

$$\Psi_{\mathcal{S}}(\{A\}, \sigma_0) \models \alpha$$

dla pewnej funkcji $\Psi_{\mathcal{S}}$.

Aby móc zdefiniować warunki stawiane funkcji przejścia w modelu, wprowadzimy kilka pojęć pomocniczych.

Definicja 29. *Zdefiniujmy pomocniczą funkcję $Res_0 : 2^{\mathcal{A}c} \times \Sigma \to 2^{\Sigma}$.*

Niech dany będzie stan $\sigma \in \Sigma$ i akcja złożona $A \subseteq \mathcal{A}c$. Niech $\Delta_1(A, \sigma), \ldots, \Delta_n(A, \sigma)$ będą dekompozycjami akcji A względem stanu σ. Wówczas stan $\sigma' \in Res_0(\sigma, A)$ wtedy i tylko wtedy, gdy σ' jest potencjalnym wynikiem wykonania A dla pewnej dekompozycji $\Delta_i(A, \sigma)$.

Funkcja Res_0 określa możliwe zmiany w stanie σ spowodowane przez zdania ***causes***.

Definicja 30. *Dla każdych stanów $\sigma, \sigma' \in \Sigma$ i dla każdej akcji $A \in \mathcal{A}c$, zbiorem $New(A, \sigma, \sigma')$ będziemy nazywać zbiór wszystkich literałów $\overline{f}$ takich, że $\sigma' \models f$ oraz:*

- *f jest inercjalny oraz $\sigma(f) \neq \sigma'(f)$, lub*
- *w $\mathcal{D}$ istnieje zdanie $(A\, \textbf{\textit{releases}}\, f\, \textbf{\textit{if}}\, \pi)$ oraz $\sigma \models \pi$.*

Zbiory New określają, które fluenty inercjalne zmieniły się pomiędzy dwoma stanami, a ponadto uwzględniają możliwość dowolnych fluktuacji fluentów uwolnionych. Jest to drugi aspekt niedeterministyczny w rozważanym systemie dynamicznym.

Definicja 31. *Niech $\mathcal{D}$ będzie domeną akcji w $\mathcal{A}c$. Struktura $\mathcal{S} = (\Sigma, \sigma_0, Res)$ jest **modelem** $\mathcal{D}$ wtw:*

1. *Σ jest zbiorem wszystkich stanów w $\mathcal{D}$ (tj. zbiorem wszystkich stanów satysfakcjonujących wszystkie ograniczenia z $\mathcal{D}$),*
2. *każde zdanie wartości i każde zdanie obserwacji jest prawdziwe w $\mathcal{S}$,*
3. *dla każdej akcji $A \subseteq \mathcal{A}c$ i dla każdego stanu $\sigma \in \Sigma$, $Res(A, \sigma) \subseteq Res_0(A, \sigma)$ jest zbiorem wszystkich stanów σ', dla których zbiory $New(A, \sigma, \sigma')$ są minimalne ze względu na relację inkluzji.*

Tak zdefiniowany model łączy niedeterminizm związany z wyborem dekompozycji akcji złożonych (funkcja Res_0) i niedeterminizm związany z fluentami uwolnionymi (zbiór New).

3 Język kwerend

W języku kwerend formułowane będą zapytania stawiane systemowi dynamicznemu, którego zadaniem jest podanie odpowiedzi zgodnych z daną domeną akcji.

3.1 Składnia języka

Język kwerend wyróżnia ich następujące rodzaje:

Definicja 32. ***Egzystencjalną kwerendą wartości*** *nazywamy zdanie*

$$\textbf{\textit{possibly}}\,\gamma\ \textbf{\textit{after}}\,(A_1, \ldots, A_n)$$

gdzie $\gamma \in Forms(f)$, $A_1, \ldots, A_n \subseteq \mathcal{A}c$.

Definicja 33. ***Ogólną kwerendą wartości*** *nazywamy zdanie*

$$\textbf{\textit{necessary}}\,\gamma\ \textbf{\textit{after}}\,(A_1, \ldots, A_n)$$

gdzie $\gamma \in Forms(f)$, $A_1, \ldots, A_n \subseteq \mathcal{A}c$.

Kwerenda egzystencjalna określa, czy wykonanie programu $(A_1, \ldots, A_n)$ ze stanu początkowego prowadzi kiedykolwiek do osiągnięcia celu γ; z kolei ogólna określa, czy wykonanie programu $(A_1, \ldots, A_n)$ ze stanu początkowego zawsze prowadzi do osiągnięciu celu γ.

Definicja 34. ***Egzystencjalną kwerendą wykonywalności*** *nazywamy zdanie*

$$\textbf{\textit{executable sometimes}}\,(A_1, \ldots, A_n)$$

gdzie $A_1, \ldots, A_n \subseteq \mathcal{A}c$.

Definicja 35. ***Ogólną kwerendą wykonywalności*** *nazywamy zdanie*

$$\textbf{\textit{executable always}}\,(A_1, \ldots, A_n)$$

gdzie $A_1, \ldots, A_n \subseteq \mathcal{A}c$.

Kwerendy wykonywalności mają na celu odpowiedzieć na pytanie, czy dany program jest kiedykolwiek (egzystencjalna) lub zawsze (ogólna) wykonywalny ze stanu początkowego.

Definicja 36. ***Kwerendą osiągalności*** *nazywamy zdanie*

$$\textbf{\textit{accessible}}\,\gamma$$

gdzie $\gamma \in Forms(f)$.

Celem tej kwerendy jest określenie, czy istnieje taki program $(A_1, \ldots, A_n)$, $n \geq 0$, wykonywalny ze stanu początkowego, po którego wykonaniu osiągnięty zostanie cel γ.

3.2 Semantyka języka

Definicja 37. *Niech $\mathcal{D}$ będzie domeną akcji, a $\mathcal{Q}$ kwerendą. Mówimy, że $\mathcal{Q}$ jest* ***konsekwencją*** *$\mathcal{D}$ (ozn. $\mathcal{D} \models\!\!\approx \mathcal{Q}$), jeżeli:*

1. *w przypadku gdy $\mathcal{Q}$ jest kwerendą wartości postaci:*
   - ***possibly*** γ ***after*** $(A_1, \ldots, A_n)$ *– wówczas $\mathcal{D} \models\!\!\approx \mathcal{Q}$ wtedy i tylko wtedy, gdy dla każdego modelu $\mathcal{S} = (\Sigma, \sigma_0, Res)$ domeny akcji $\mathcal{D}$ istnieje funkcja $\Psi_{\mathcal{S}}$ taka, że zachodzi*
   $$\Psi_S((A_1, ..., A_n), \sigma_0) \models \gamma$$
   - ***necessary*** γ ***after*** $(A_1, \ldots, A_n)$ *– wówczas $\mathcal{D} \models\!\!\approx \mathcal{Q}$ wtedy i tylko wtedy, gdy dla każdego modelu $\mathcal{S} = (\Sigma, \sigma_0, Res)$ domeny akcji $\mathcal{D}$ i dla każdej funkcji $\Psi_{\mathcal{S}}$ zachodzi*
   $$\Psi_{\mathcal{S}}((A_1, ..., A_n), \sigma_0) \models \gamma$$
2. *w przypadku gdy $\mathcal{Q}$ jest kwerendą wykonywalności postaci:*
   - ***executable sometimes*** $(A_1, \ldots, A_n)$ *– wówczas $\mathcal{D} \models\!\!\approx \mathcal{Q}$ wtedy i tylko wtedy, gdy dla każdego modelu $\mathcal{S} = (\Sigma, \sigma_0, Res)$ domeny akcji $\mathcal{D}$ takiego, że istnieje funkcja $\Psi_{\mathcal{S}}$ taka, że*
   $$\Psi_{\mathcal{S}}((A_1, ..., A_n), \sigma_0)$$
   jest zdefiniowane,
   - ***executable always*** $(A_1, \ldots, A_n)$ *– wówczas $\mathcal{D} \models\!\!\approx \mathcal{Q}$ wtedy i tylko wtedy, gdy dla każdego modelu $\mathcal{S} = (\Sigma, \sigma_0, Res)$ domeny akcji $\mathcal{D}$ i dla każdej funkcji $\Psi_{\mathcal{S}}$*
   $$\Psi_{\mathcal{S}}((A_1, ..., A_n), \sigma_0)$$
   jest zdefiniowane,
3. *w przypadku gdy $\mathcal{Q}$ jest kwerendą osiągalności postaci* ***accessible*** γ *– wówczas $\mathcal{D} \models\!\!\approx \mathcal{Q}$ wtedy i tylko wtedy, gdy dla każdego modelu $\mathcal{S} = (\Sigma, \sigma_0, Res)$ domeny akcji $\mathcal{D}$, dla każdej funkcji $\Psi_{\mathcal{S}}$ istnieje ciąg akcji $(A_1, ..., A_n)$, $n \geq 0$ taki, że:*
   - $\Psi_{\mathcal{S}}((A_1, ..., A_n), \sigma_0)$ *jest zdefiniowane,*
   - $\Psi_{\mathcal{S}}((A_1, ..., A_n), \sigma_0) \models \gamma$.

4 Przykłady

4.1 Producent i konsument

Załóżmy, że dany jest producent i konsument. Producent może wstawić do bufora dokładnie jeden produkt przy pomocy akcji PUT. Konsument może pobrać z bufora produkt z użyciem akcji GET, oraz pozbyć się go z użyciem akcji CONSUME. Konsument nie może skonsumować produktu, jeśli wcześniej go nie pobrał. Na początku bufor jest pusty.

Reprezentacja tego scenariusza w postaci domeny akcji $\mathcal{D}$ wygląda następująco:

$$\begin{aligned}
&\textsc{Put} \ \textbf{\textit{causes}} \ \neg \mathit{bufferEmpty} \ \textbf{\textit{if}} \ \mathit{bufferEmpty} \\
&\textsc{Get} \ \textbf{\textit{causes}} \ \mathit{bufferEmpty} \wedge \mathit{hasItem} \ \textbf{\textit{if}} \ \neg \mathit{bufferEmpty} \wedge \neg \mathit{hasItem} \\
&\textsc{Consume} \ \textbf{\textit{causes}} \ \neg \mathit{hasItem} \\
&\textbf{\textit{impossible}} \ \textsc{Consume} \ \textbf{\textit{if}} \ \neg \mathit{hasItem} \\
&\textbf{\textit{initially}} \ \mathit{bufferEmpty}
\end{aligned}$$

Rozważmy po kolei wykonanie wszystkich możliwych akcji złożonych w każdym stanie. Możliwe są cztery stany:

1. $\sigma_0 = \{\mathit{bufferEmpty}, \mathit{hasItem}\}$,
2. $\sigma_1 = \{\neg \mathit{bufferEmpty}, \mathit{hasItem}\}$,
3. $\sigma_2 = \{\mathit{bufferEmpty}, \neg \mathit{hasItem}\}$,
4. $\sigma_3 = \{\neg \mathit{bufferEmpty}, \neg \mathit{hasItem}\}$.

Potencjalnymi stanami początkowymi są σ_0 oraz σ_2.

- Dla stanu σ_0 mamy

$$\begin{aligned}
New(\sigma_0, \sigma_0) &= \emptyset \\
New(\sigma_0, \sigma_1) &= \{\neg \mathit{bufferEmpty}\} \\
New(\sigma_0, \sigma_2) &= \{\neg \mathit{hasItem}\} \\
New(\sigma_0, \sigma_3) &= \{\neg \mathit{bufferEmpty}, \neg \mathit{hasItem}\}
\end{aligned}$$

$$\begin{aligned}
Res_0(\sigma_0, \{\textsc{Put}\}) &= \{\sigma_1, \sigma_3\} \\
Res_0(\sigma_0, \{\textsc{Put}\}) &= \{\sigma_1\}
\end{aligned}$$

$$\begin{aligned}
Res_0(\sigma_0, \{\textsc{Get}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_0, \{\textsc{Get}\}) &= \{\sigma_0\}
\end{aligned}$$

$$\begin{aligned}
Res_0(\sigma_0, \{\textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_0, \{\textsc{Consume}\}) &= \{\sigma_2\}
\end{aligned}$$

$$\begin{aligned}
Res_0(\sigma_0, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_1, \sigma_3\} \\
Res(\sigma_0, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_1\}
\end{aligned}$$

$$Res_0(\sigma_0, \{\textsc{Put}, \textsc{Consume}\}) = \{\sigma_3\} = Res(\sigma_0, \{\textsc{Put}, \textsc{Consume}\})$$

$$\begin{aligned}
Res_0(\sigma_0, \{\textsc{Get}, \textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_0, \{\textsc{Get}, \textsc{Consume}\}) &= \{\sigma_2\}
\end{aligned}$$

$$Res_0(\sigma_0, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\}) = \{\sigma_3\} = Res(\sigma_0, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\})$$

- Dla stanu σ_1 mamy

$$
\begin{aligned}
New(\sigma_1, \sigma_0) &= \{bufferEmpty\} \\
New(\sigma_1, \sigma_1) &= \emptyset \\
New(\sigma_1, \sigma_2) &= \{bufferEmpty, \neg hasItem\} \\
New(\sigma_1, \sigma_3) &= \{\neg hasItem\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Get}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Get}\}) &= \{\sigma_1\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Put}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Put}\}) &= \{\sigma_1\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Consume}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_1\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Put}, \textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Put}, \textsc{Consume}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Get}, \textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Get}, \textsc{Consume}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\}) &= \{\sigma_2, \sigma_3\} \\
Res(\sigma_1, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\}) &= \{\sigma_3\}
\end{aligned}
$$

- Dla stanu σ_2 mamy

$$
\begin{aligned}
New(\sigma_2, \sigma_0) &= \{hasItem\} \\
New(\sigma_2, \sigma_1) &= \{\neg bufferEmpty, hasItem\} \\
New(\sigma_2, \sigma_2) &= \emptyset \\
New(\sigma_2, \sigma_3) &= \{\neg bufferEmpty\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_2, \{\textsc{Get}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_2, \{\textsc{Get}\}) &= \{\sigma_2\}
\end{aligned}
$$

$$
\begin{aligned}
Res_0(\sigma_2, \{\textsc{Put}\}) &= \{\sigma_1, \sigma_3\} \\
Res(\sigma_2, \{\textsc{Put}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$Res_0(\sigma_2, \{\textsc{Consume}\}) = \emptyset = Res(\sigma_2, \{\textsc{Consume}\})$$

$$
\begin{aligned}
Res_0(\sigma_2, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_1, \sigma_3\} \\
Res(\sigma_2, \{\textsc{Put}, \textsc{Get}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$Res_0(\sigma_2, \{\textsc{Put}, \textsc{Consume}\}) = \emptyset = Res(\sigma_2, \{\textsc{Put}, \textsc{Consume}\})$$

$$Res_0(\sigma_2, \{\textsc{Get}, \textsc{Consume}\}) = \emptyset = Res(\sigma_2, \{\textsc{Get}, \textsc{Consume}\})$$

$$Res_0(\sigma_2, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\}) = \emptyset = Res(\sigma_2, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\})$$

Występowanie zbiorów pustych w akcjach złożonych zawierających akcję atomową Consume zapewnia definicja 21 – nie istnieje żaden potencjalny wynik wykonania tych akcji.

- Dla stanu σ_3 mamy

$$
\begin{aligned}
New(\sigma_3, \sigma_0) &= \{bufferEmpty, hasItem\} \\
New(\sigma_3, \sigma_1) &= \{\neg hasItem\} \\
New(\sigma_3, \sigma_2) &= \{bufferEmpty\} \\
New(\sigma_3, \sigma_3) &= \emptyset
\end{aligned}
$$

$$Res_0(\sigma_3, \{\textsc{Get}\}) = \{\sigma_0\} = Res(\sigma_3, \{\textsc{Get}\})$$

$$
\begin{aligned}
Res_0(\sigma_3, \{\textsc{Put}\}) &= \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} \\
Res(\sigma_3, \{\textsc{Put}\}) &= \{\sigma_3\}
\end{aligned}
$$

$$Res_0(\sigma_3, \{\textsc{Consume}\}) = \emptyset = Res(\sigma_3, \{\textsc{Consume}\})$$

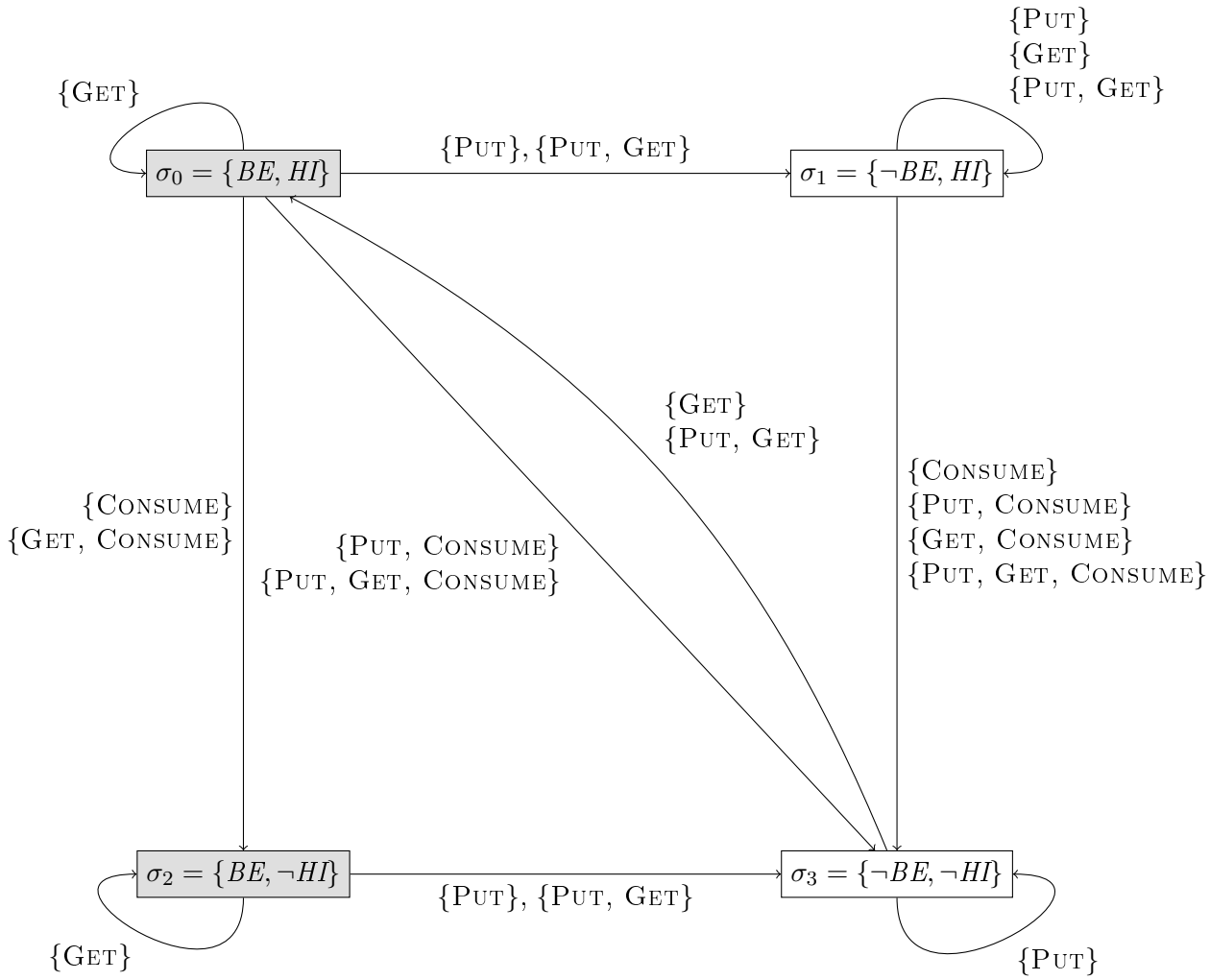
$$Res_0(\sigma_3, \{\textsc{Put}, \textsc{Get}\}) = \{\sigma_0\} = Res(\sigma_3, \{\textsc{Put}, \textsc{Get}\})$$

$$Res_0(\sigma_3, \{\textsc{Put}, \textsc{Consume}\}) = \emptyset = Res(\sigma_3, \{\textsc{Put}, \textsc{Consume}\})$$

$$Res_0(\sigma_3, \{\textsc{Get}, \textsc{Consume}\}) = \emptyset = Res(\sigma_3, \{\textsc{Get}, \textsc{Consume}\})$$

$$Res_0(\sigma_3, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\}) = \emptyset = Res(\sigma_3, \{\textsc{Put}, \textsc{Get}, \textsc{Consume}\})$$

Warto zauważyć, że w tym przypadku funkcja przejścia jest całkowicie deterministyczna, więc istnieje dokładnie jedna funkcja $\Psi_{\mathcal{S}}$ (choć jest ona nadal częściowo zdefiniowana). Graf funkcji przejścia dla tego scenariusza znajduje się na rysunku 2.



Rysunek 1: Graf funkcji przejścia dla scenariusza *producent-konsument*. Kolorem szarym oznaczone potencjalne stany początkowe.

Rozważmy teraz kilka przykładowych kwerend nad $\mathcal{D}$.

1. Niech dana będzie kwerenda

$$\textbf{\textit{possibly}}\ \neg \mathit{bufferEmpty} \wedge \neg \mathit{hasItem}\ \textbf{\textit{after}}\ (\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$$

   Kwerenda ta **nie będzie** konsekwencją $\mathcal{D}$, ponieważ w modelu $\mathcal{D}$, którego stanem początkowym jest σ_2, program $(\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$ nie jest wykonywalny.

2. Niech dana będzie kwerenda

$$\textbf{\textit{necessary}}\ \neg \mathit{bufferEmpty} \wedge \neg \mathit{hasItem}\ \textbf{\textit{after}}\ (\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$$

   Kwerenda ta **nie będzie** konsekwencją $\mathcal{D}$, ponieważ w modelu $\mathcal{D}$, którego stanem początkowym jest σ_2, program $(\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$ nie jest wykonywalny, więc odpowiednia wartość $\Psi_{\mathcal{S}}$ w definicji konsekwencji nie jest zdefiniowana.

3. Analogicznie, kwerendy

$$\textbf{\textit{executable sometimes}}\ (\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$$
$$\textbf{\textit{executable always}}\ (\{\textsc{Put}\}, \{\textsc{Get}, \textsc{Consume}\})$$

   nie są konsekwencją $\mathcal{D}$.

4. Niech dana będzie kwerenda

$$\textbf{\textit{accessible}}\ \mathit{bufferEmpty} \vee \neg \mathit{hasItem}$$

   Kwerenda ta **jest** konsekwencją $\mathcal{D}$, ponieważ w szczególności program pusty prowadzi do spełnienia celu $\mathit{bufferEmpty} \vee \neg \mathit{hasItem}$ (tj. w każdym modelu stan początkowy spełnia zadany cel).

4.2 Kot Schrödingera

Załóżmy, że dany jest pojemnik z kotem i trucizną, która w każdym momencie może zostać uwolniona. Na początku wiadomo, że kot żyje. Przy pomocy akcji Peek można zajrzeć do pojemnika, po czym może się okazać, że kot umarł, zaczął lub przestał mruczeć. Jeżeli kot żyje, wykonanie akcji Pet spowoduje, że kot zacznie mruczeć. Jeżeli kot nie żyje, niemożliwe jest żeby mruczał.

Powyższy scenariusz jest reprezentowany przez następującą domenę akcji $\mathcal{D}$:

$$\textbf{\textit{initially}}\ \mathit{alive}$$
$$\textsc{Peek}\ \textbf{\textit{releases}}\ \mathit{alive}\ \textbf{\textit{if}}\ \mathit{alive}$$
$$\textsc{Peek}\ \textbf{\textit{releases}}\ \mathit{purring}\ \textbf{\textit{if}}\ \mathit{alive}$$
$$\textsc{Pet}\ \textbf{\textit{causes}}\ \mathit{purring}\ \textbf{\textit{if}}\ \mathit{alive}$$
$$\textbf{\textit{always}}\ \neg \mathit{alive} \rightarrow \neg \mathit{purring}$$

Rozważmy po kolei wykonanie wszystkich możliwych akcji złożonych w każdym stanie. Możliwe są trzy stany:

1. $\sigma_0 = \{\mathit{alive}, \mathit{purring}\}$,
2. $\sigma_1 = \{\mathit{alive}, \neg \mathit{purring}\}$,
3. $\sigma_2 = \{\neg \mathit{alive}, \neg \mathit{purring}\}$.

Potencjalnymi stanami początkowymi są σ_0 oraz σ_1.

- Dla stanu σ_0 mamy

$$
\begin{aligned}
Res_0(\sigma_0, \{\textsc{Peek}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\
New(\{\textsc{Peek}\}, \sigma_0, \sigma_0) &= \{alive, purring\} \\
New(\{\textsc{Peek}\}, \sigma_0, \sigma_1) &= \{alive, \neg purring\} \\
New(\{\textsc{Peek}\}, \sigma_0, \sigma_2) &= \{\neg alive, \neg purring\} \\
Res(\sigma_0, \{\textsc{Peek}\}) &= \{\sigma_0, \sigma_1, \sigma_2\}
\end{aligned}
$$

$$
Res_0(\sigma_0, \{\textsc{Pet}\}) = \{\sigma_0\} = Res(\sigma_0, \{\textsc{Pet}\})
$$

$$
\begin{aligned}
\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0) &= \{\textsc{Peek}\} \\
\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0) &= \{\textsc{Pet}\} \\
Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0), \sigma_0) &= Res_0(\sigma_0, \{\textsc{Peek}\}) \\
Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0), \sigma_0) &= Res_0(\sigma_0, \{\textsc{Pet}\}) \\
Res_0(\sigma_0, \{\textsc{Peek}, \textsc{Pet}\}) &= Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0), \sigma_0) \cup Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0), \sigma_0) \\
Res_0(\sigma_0, \{\textsc{Peek}, \textsc{Pet}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0, \sigma_0) &= \{alive, purring\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0, \sigma_1) &= \{alive, \neg purring\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_0, \sigma_2) &= \{\neg alive, \neg purring\} \\
Res(\sigma_0, \{\textsc{Peek}, \textsc{Pet}\}) &= \{\sigma_0, \sigma_1, \sigma_2\}
\end{aligned}
$$

  W stanie σ_0 akcje $\textsc{Peek}, \textsc{Pet}$ są konfliktowe, ponieważ obie wpływają na fluent *purring*. Dlatego dla akcji złożonej $\{\textsc{Peek}, \textsc{Pet}\}$ wyróżnione są dwie dekompozycje Δ_1 i Δ_2.

- Dla stanu σ_1 mamy

$$
\begin{aligned}
Res_0(\sigma_1, \{\textsc{Peek}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\
New(\{\textsc{Peek}\}, \sigma_1, \sigma_0) &= \{alive, purring\} \\
New(\{\textsc{Peek}\}, \sigma_1, \sigma_1) &= \{alive, \neg purring\} \\
New(\{\textsc{Peek}\}, \sigma_1, \sigma_2) &= \{\neg alive, \neg purring\} \\
Res(\sigma_1, \{\textsc{Peek}\}) &= \{\sigma_0, \sigma_1, \sigma_2\}
\end{aligned}
$$

$$
Res_0(\sigma_1, \{\textsc{Pet}\}) = \{\sigma_0\} = Res(\sigma_1, \{\textsc{Pet}\})
$$

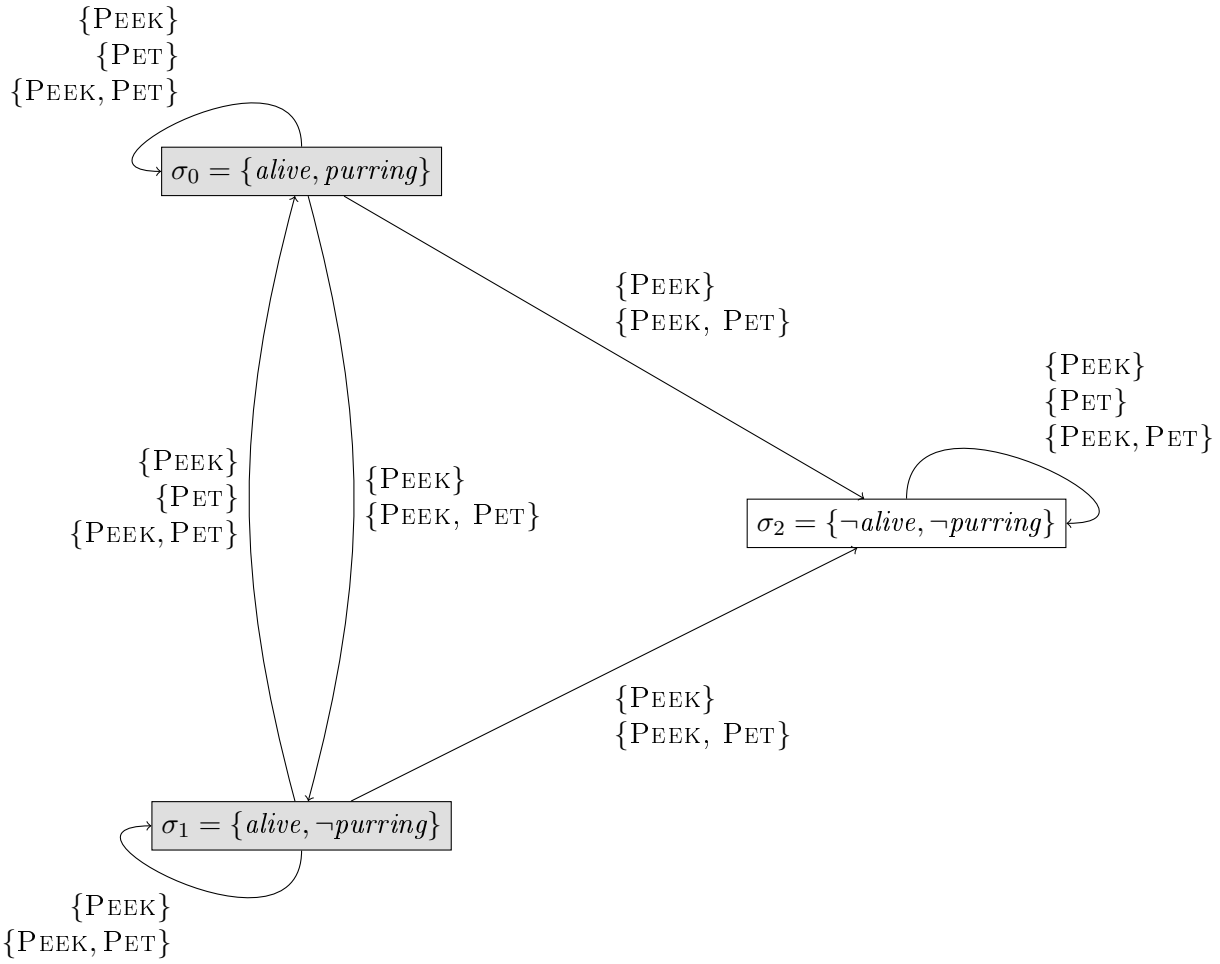
$$
\begin{aligned}
\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1) &= \{\textsc{Peek}\} \\
\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1) &= \{\textsc{Pet}\} \\
Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1), \sigma_1) &= Res_0(\sigma_1, \{\textsc{Peek}\}) \\
Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1), \sigma_1) &= Res_0(\sigma_1, \{\textsc{Pet}\}) \\
Res_0(\sigma_1, \{\textsc{Peek}, \textsc{Pet}\}) &= Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1), \sigma_1) \cup Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1), \sigma_1) \\
Res_0(\sigma_1, \{\textsc{Peek}, \textsc{Pet}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1, \sigma_0) &= \{alive, purring\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1, \sigma_1) &= \{alive, \neg purring\} \\
New(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_1, \sigma_2) &= \{\neg alive, \neg purring\} \\
Res(\sigma_1, \{\textsc{Peek}, \textsc{Pet}\}) &= \{\sigma_0, \sigma_1, \sigma_2\}
\end{aligned}
$$

- Dla stanu σ_2 mamy

$$Res_0(\sigma_2, \{\textsc{Peek}\}) = \{\sigma_2\} = Res(\sigma_2, \{\textsc{Peek}\})$$

$$Res_0(\sigma_2, \{\textsc{Pet}\}) = \{\sigma_2\} = Res(\sigma_2, \{\textsc{Pet}\})$$

$$\begin{aligned}
\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2) &= \{\textsc{Peek}\} \\
\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2) &= \{\textsc{Pet}\} \\
Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2), \sigma_2) &= Res_0(\sigma_2, \{\textsc{Peek}\}) \\
Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2), \sigma_2) &= Res_0(\sigma_2, \{\textsc{Pet}\}) \\
Res_0(\sigma_2, \{\textsc{Peek}, \textsc{Pet}\}) &= Res_0(\Delta_1(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2), \sigma_2) \cup Res_0(\Delta_2(\{\textsc{Peek}, \textsc{Pet}\}, \sigma_2), \sigma_2) \\
Res_0(\sigma_2, \{\textsc{Peek}, \textsc{Pet}\}) &= \{\sigma_2\} = Res(\sigma_2, \{\textsc{Peek}, \textsc{Pet}\})
\end{aligned}$$



Rysunek 2: Graf funkcji przejścia dla scenariusza *Kot Schrödingera*. Kolorem szarym oznaczone potencjalne stany początkowe.

Przykładowe kwerendy nad $\mathcal{D}$:

1. Niech dana będzie kwerenda

   $$\textbf{\textit{possibly}}\ \neg alive\ \textbf{\textit{after}}\ (\{\textsc{Pet}\}, \{\textsc{Peek}\})$$

   Kwerenda ta **będzie** konsekwencją $\mathcal{D}$, ponieważ dla modelu $\mathcal{D}$, którego stanem początkowym jest σ_0 i dla modelu którego stanem początkowym jest σ_1, istnieje $\Psi_{\mathcal{S}}$, że program $(\{\textsc{Pet}\}, \{\textsc{Peek}\})$ prowadzi do stanu σ_2, w którym spełnione jest $\neg alive$.

2. Niech dana będzie kwerenda

   $$\textbf{\textit{necessary}}\ purring\ \textbf{\textit{after}}\ (\{\textsc{Peek}, \textsc{Pet}\})$$

   Kwerenda ta **nie będzie** konsekwencją $\mathcal{D}$, ponieważ w modelu $\mathcal{D}$, którego stanem początkowym jest σ_0 lub σ_1, dla programu $(\{\textsc{Peek}, \textsc{Pet}\})$ istnieje ścieżka $\Psi_{\mathcal{S}}$ prowadząca do stanu σ_2.

3. Kwerendy ***executable sometimes*** oraz ***executable always*** **są** konsekwencjami $\mathcal{D}$ dla każdego programu $\mathcal{P}$, ponieważ dowolny program da się wykonać dla każdego modelu $\mathcal{D}$ z dowolnym stanem początkowym, ze względu na brak zdań ***impossible***.

4. Niech dana będzie kwerenda

   $$\textbf{\textit{accessible}}\ purring$$

   Kwerenda ta **jest** konsekwencją $\mathcal{D}$, ponieważ dla modelu $\mathcal{D}$, którego stanem początkowym jest σ_0, pusty program prowadzi do osiągnięcia celu. Z kolei dla modelu, którego stanem poczatkowym jest σ_1, program $(\{\textsc{Pet}\})$ prowadzi do osiągnięcia ceul.

4.3 Dwóch malarzy

Załóżmy, że mamy dwóch malarzy (oznaczmy ich jako A i B) i jeden pędzel. Malarze mogą zabrać pędzel (akcje $\textsc{TakeA}, \textsc{TakeB}$), na początku żaden z malarzy nie ma pędzla. Malarze mający pędzel mogą wykonać akcję $\textsc{Paint}$, po której znów odkładają pędzel.

Powyższy scenariusz jest reprezentowany przez następującą domenę akcji $\mathcal{D}$:

$$\begin{aligned}
&\textbf{\textit{initially}}\ \neg brushA, \neg brushB\\
&\textsc{TakeA}\ \textbf{\textit{causes}}\ brushA \wedge \neg brushB\\
&\textsc{TakeB}\ \textbf{\textit{causes}}\ \neg brushA \wedge brushB\\
&\textsc{Paint}\ \textbf{\textit{causes}}\ \neg brushA\ \textbf{\textit{if}}\ brushA\\
&\textsc{Paint}\ \textbf{\textit{causes}}\ \neg brushB\ \textbf{\textit{if}}\ brushB\\
&\textbf{\textit{always}}\ \neg brushA \vee \neg brushB
\end{aligned}$$

Rozważmy po kolei wykonanie wszystkich możliwych akcji złożonych w każdym stanie. Możliwe są trzy stany:

1. $\sigma_0 = \{\neg brushA, \neg brushB\}$,
2. $\sigma_1 = \{brushA, \neg brushB\}$,

3. $\sigma_2 = \{\neg brushA, brushB\}$.

Stanem początkowym będzie σ_0.

- Dla stanu σ_0 mamy

$$Res_0(\sigma_0, \{\textsc{TakeA}\}) = \{\sigma_1\} = Res(\sigma_0, \{\textsc{TakeA}\})$$

$$Res_0(\sigma_0, \{\textsc{TakeB}\}) = \{\sigma_2\} = Res(\sigma_0, \{\textsc{TakeB}\})$$

$$Res_0(\sigma_0, \{\textsc{Paint}\}) = \{\sigma_0\} = Res(\sigma_0, \{\textsc{Paint}\})$$

$$\begin{aligned}
Res_0(\sigma_0, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_0\})) &= Res_0(\sigma_0, \{\textsc{TakeA}\}) = \{\sigma_1\} \\
Res_0(\sigma_0, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_0\})) &= Res_0(\sigma_0, \{\textsc{TakeB}\}) = \{\sigma_2\} \\
Res_0(\sigma_0, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_1, \sigma_2\} \\
New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_0, \sigma_1) &= \{brushA\} \\
New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_0, \sigma_2) &= \{brushB\} \\
Res(\sigma_0, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_1, \sigma_2\}
\end{aligned}$$

$$Res_0(\sigma_0, \{\textsc{TakeA},\ \textsc{Paint}\}) = \{\sigma_1\} = Res(\sigma_0, \{\textsc{TakeA},\ \textsc{Paint}\})$$

$$Res_0(\sigma_0, \{\textsc{TakeB},\ \textsc{Paint}\}) = \{\sigma_2\} = Res(\sigma_0, \{\textsc{TakeB},\ \textsc{Paint}\})$$

$$\begin{aligned}
Res_0(\sigma_0, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_0\})) &= Res_0(\sigma_0, \{\textsc{TakeA},\ \textsc{Paint}\}) = \{\sigma_1\} \\
Res_0(\sigma_0, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_0\})) &= Res_0(\sigma_0, \{\textsc{TakeB},\ \textsc{Paint}\}) = \{\sigma_2\} \\
Res_0(\sigma_0, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_1, \sigma_2\} \\
New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_0, \sigma_1) &= \{brushA\} \\
New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_0, \sigma_2) &= \{brushB\} \\
Res(\sigma_0, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_1, \sigma_2\}
\end{aligned}$$

W tym stanie akcje TakeA i TakeB są konfliktowe, fluenty *brushA, brushB* są sprzeczne, dlatego dla akcji zawierających TakeA i TakeB trzeba wyznaczyć dekompozycje.

- Dla stanu σ_1 mamy

$$Res_0(\sigma_1, \{\textsc{TakeA}\}) = \{\sigma_1\} = Res(\sigma_1, \{\textsc{TakeA}\})$$

$$Res_0(\sigma_1, \{\textsc{TakeB}\}) = \{\sigma_2\} = Res(\sigma_1, \{\textsc{TakeB}\})$$

$$\begin{aligned} Res_0(\sigma_1, \{\textsc{Paint}\}) &= \{\sigma_0, \sigma_2\} \\ New(\{\textsc{Paint}\}, \sigma_1, \sigma_0) &= \{brushA\} \\ New(\{\textsc{Paint}\}, \sigma_1, \sigma_2) &= \{brushA, brushB\} \\ Res(\sigma_1, \{\textsc{Paint}\}) &= \{\sigma_0\} \end{aligned}$$

$$\begin{aligned} Res_0(\sigma_1, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{TakeA}\}) = \{\sigma_1\} \\ Res_0(\sigma_1, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{TakeB}\}) = \{\sigma_2\} \\ Res_0(\sigma_1, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_1, \sigma_2\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_1, \sigma_1) &= \emptyset \\ New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_1, \sigma_2) &= \{brushA, brushB\} \\ Res(\sigma_0, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_1\} \end{aligned}$$

$$\begin{aligned} Res_0(\sigma_1, \Delta_1(\{\textsc{TakeA},\ \textsc{Paint}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{TakeA}\}) = \{\sigma_1\} \\ Res_0(\sigma_1, \Delta_2(\{\textsc{TakeA},\ \textsc{Paint}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{Paint}\}) = \{\sigma_0, \sigma_2\} \\ Res_0(\sigma_1, \{\textsc{TakeA},\ \textsc{Paint}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\ New(\{\textsc{TakeA},\ \textsc{Paint}\}, \sigma_1, \sigma_0) &= \{brushA\} \\ New(\{\textsc{TakeA},\ \textsc{Paint}\}, \sigma_1, \sigma_1) &= \emptyset \\ New(\{\textsc{TakeA},\ \textsc{Paint}\}, \sigma_1, \sigma_2) &= \{brushA, brushB\} \\ Res(\sigma_1, \{\textsc{TakeA},\ \textsc{Paint}\}) &= \{\sigma_1\} \end{aligned}$$

$$Res_0(\sigma_1, \{\textsc{TakeB},\ \textsc{Paint}\})) = \{\sigma_2\} = Res(\sigma_1, \{\textsc{TakeB},\ \textsc{Paint}\}))$$

$$\begin{aligned} Res_0(\sigma_1, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{TakeA}\}) = \{\sigma_1\} \\ Res_0(\sigma_1, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_1\})) &= Res_0(\sigma_1, \{\textsc{TakeB},\ \textsc{Paint}\}) = \{\sigma_2\} \\ Res_0(\sigma_1, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_1, \sigma_2\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_1, \sigma_1) &= \emptyset \\ New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_1, \sigma_2) &= \{brushA, brushB\} \\ Res(\sigma_1, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_1\} \end{aligned}$$

W stanie σ_1 dodatkowo TakeA, Paint są sprzeczne, ustawiają fluent *BrushA* na przeciwne wartości.

- Dla stanu σ_2 mamy

$$Res_0(\sigma_2, \{\textsc{TakeA}\}) = \{\sigma_1\} = Res(\sigma_2, \{\textsc{TakeA}\})$$

$$Res_0(\sigma_2, \{\textsc{TakeB}\}) = \{\sigma_2\} = Res(\sigma_2, \{\textsc{TakeB}\})$$

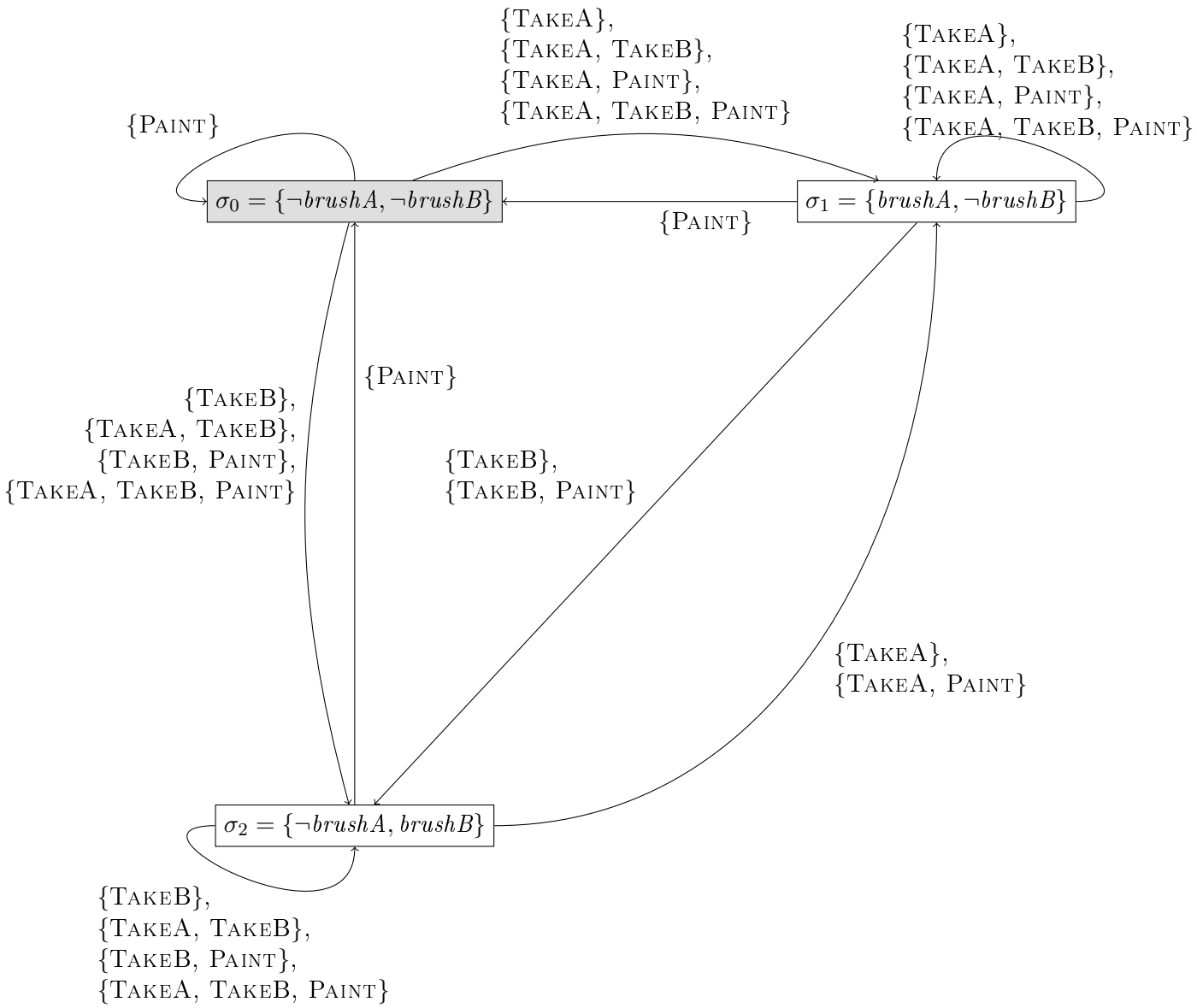
$$\begin{aligned} Res_0(\sigma_2, \{\textsc{Paint}\}) &= \{\sigma_0, \sigma_1\} \\ New(\{\textsc{Paint}\}, \sigma_2, \sigma_0) &= \{brushB\} \\ New(\{\textsc{Paint}\}, \sigma_2, \sigma_1) &= \{brushA, brushB\} \\ Res(\sigma_2, \{\textsc{Paint}\}) &= \{\sigma_0\} \end{aligned}$$

$$\begin{aligned} Res_0(\sigma_2, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_2\})) &= Res_0(\sigma_1, \{\textsc{TakeA}\}) = \{\sigma_1\} \\ Res_0(\sigma_2, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_2\})) &= Res_0(\sigma_1, \{\textsc{TakeB}\}) = \{\sigma_2\} \\ Res_0(\sigma_2, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_1, \sigma_2\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_2, \sigma_1) &= \{brushA, brushB\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB}\}, \sigma_2, \sigma_2) &= \emptyset \\ Res(\sigma_2, \{\textsc{TakeA},\ \textsc{TakeB}\}) &= \{\sigma_2\} \end{aligned}$$

$$Res_0(\sigma_2, \{\textsc{TakeA},\ \textsc{Paint}\})) = \{\sigma_1\} = Res(\sigma_2, \{\textsc{TakeA},\ \textsc{Paint}\}))$$

$$\begin{aligned} Res_0(\sigma_2, \Delta_1(\{\textsc{TakeB},\ \textsc{Paint}\}, \sigma_2\})) &= Res_0(\sigma_2, \{\textsc{TakeB}\}) = \{\sigma_2\} \\ Res_0(\sigma_2, \Delta_2(\{\textsc{TakeB},\ \textsc{Paint}\}, \sigma_2\})) &= Res_0(\sigma_2, \{\textsc{Paint}\}) = \{\sigma_0, \sigma_1\} \\ Res_0(\sigma_2, \{\textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\ New(\{\textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_0) &= \{brushB\} \\ New(\{\textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_1) &= \{brushA, brushB\} \\ New(\{\textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_2) &= \emptyset \\ Res(\sigma_2, \{\textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_2\} \end{aligned}$$

$$\begin{aligned} Res_0(\sigma_2, \Delta_1(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_2\})) &= Res_0(\sigma_2, \{\textsc{TakeA},\ \textsc{Paint}\}) = \{\sigma_1\} \\ Res_0(\sigma_2, \Delta_2(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_2\})) &= Res_0(\sigma_2, \{\textsc{TakeB}\}) = \{\sigma_2\} \\ Res_0(\sigma_2, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_0, \sigma_1, \sigma_2\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_0) &= \{brushB\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_1) &= \{brushA, brushB\} \\ New(\{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}, \sigma_2, \sigma_2) &= \emptyset \\ Res(\sigma_2, \{\textsc{TakeA},\ \textsc{TakeB},\ \textsc{Paint}\}) &= \{\sigma_2\} \end{aligned}$$



Rysunek 3: Graf funkcji przejścia dla scenariusza *Dwóch malarzy*. Kolorem szarym oznaczone potencjalne stany początkowe.

Przykładowe kwerendy nad $\mathcal{D}$:

1. Niech dana będzie kwerenda

$$\textbf{\textit{possibly}}\ brushA\ \textbf{\textit{after}}\ (\{\textsc{TakeA}, \textsc{TakeB}\})$$

   Kwerenda ta **będzie** konsekwencją $\mathcal{D}$, ponieważ dla modelu $\mathcal{D}$, którego stanem początkowym jest σ_0, istnieje $\Psi_{\mathcal{S}}$, że program $(\{\textsc{TakeA}, \textsc{TakeB}\})$ prowadzi do stanu σ_1, w którym spełnione jest $brushA$.

2. Niech dana będzie kwerenda

$$\textbf{\textit{necessary}}\ \neg brushA \wedge \neg brushB\ \textbf{\textit{after}}\ (\{\textsc{TakeA}, \textsc{TakeB}\}, \{\textsc{TakeA}, \textsc{TakeB}, \textsc{Paint}\})$$

Kwerenda ta **nie będzie** konsekwencją $\mathcal{D}$, ponieważ w modelu $\mathcal{D}$, którego stanem początkowym jest σ_0, dla programu $(\{\text{TAKEA}, \text{TAKEB}\}, \{\text{TAKEA}, \text{TAKEB}, \text{PAINT}\})$ nie istnieje ścieżka $\Psi_{\mathcal{S}}$ prowadząca do stanu σ_0.

3. Kwerendy ***executable sometimes*** oraz ***executable always*** **są** konsekwencjami $\mathcal{D}$ dla każdego programu $\mathcal{P}$, ponieważ dowolny program da się wykonać dla modelu $\mathcal{D}$ ze stanem początkowym σ_0, ze względu na brak zdań ***impossible***.

4. Niech dana będzie kwerenda

$$\textbf{\textit{accessible}}\ brushA \wedge brushB$$

Kwerenda ta **nie będzie** konsekwencją $\mathcal{D}$, ponieważ dla modelu $\mathcal{D}$, którego stanem początkowym jest σ_0, nie istnieje program, który prowadzi do osiągnięcia celu. Cel jest sprzeczny ze zdefiniowaną domeną akcji $\mathcal{D}$.

5 Podział prac

- **Bartłomiej Dach** przygotował składnię języka kwerend (podrozdział 3.1), definicje 16 (funkcji pomocniczej Res^*), 27 i 28 (prawdziwości zdań wartości i obserwacji), 30 (zbioru New), opracował przykład 4.1 (o producencie i konsumencie) i dokonał redakcji końcowej wersji dokumentu.

- **Tymon Felski** opracował semantykę języka kwerend (podrozdział 3.2), opracował przykłady kwerend w przykładzie 4.3 i dokonał weryfikacji przykładów kwerend w pozostałych dwóch przykładach.

- **Filip Grajek** przygotował składnię języka akcji (podrozdział 2.1), definicje: 20 (konfliktu), 21 (potencjalnej wykonywalności), 22 (dekompozycji), 23 (komponentu wykonania), 24 (potencjalnego wyniku wykonania) i 29 (funkcji Res_0) oraz opracował przykład 4.3 (o dwóch malarzach).

- **Maciej Grzeszczak** przygotował definicje: 13 (domeny akcji), 14 (stanu), 15 (struktury), 25 (fluentu inercjalnego) i 26 (stanu względem domeny akcji) oraz opracował przykład 4.2 (o kocie Schrödingera).

- **Jacek Dziwulski** przygotował definicje: 18 (dysjunkcyjnej postaci normalnej) i 19 (komponentu formuły) i sformułował twierdzenie 1 (o redukcji dowolnej formuły logicznej do dysjunkcyjnej postaci normalnej).

- **Michał Kołodziej** przygotował definicje: 17 (funkcji $\Psi_{\mathcal{S}}$) i 31 (modelu).

- **Jędrzej Fijałkowski** i **Piotr Wolski** oraz pozostali wyżej wymienieni członkowie zespołu byli obecni na spotkaniach związanych z realizacją projektu.