Realizacja programów działań w środowisku wieloagentowym

Grupa 2: Piotr Konowrocki, Urszula Koc, Dawid Łazuk, Jakub Mazurkiewicz, Michał Omelańczuk, Bartek Polak, Arek Ryszewski, Kacper Zegadło

3 maja 2019

1 Założenia

Podstawowymi założeniami klasy systemów dynamicznych opisanych w naszym projekcie są:

- Prawo inercji.
- Sekwencyjność działań.
- Niedeterminizm działań.
- Warunki integralności wpływają tylko na skutki pośrednie.
- Z każdą akcją związany jest warunek początkowy π, zbiór G agentów (wykonawców akcji) oraz efekt akcji α zależny od warunku początkowego akcji i wykonawców tej akcji.
- W pewnych stanach działania mogą być niewykonalne przez wszystkich lub wskazanych agentów.

Do reprezentacji takiej klasy systemów dynamicznych nasz zespół opracował klasę języków akcji $\mathcal{MAR}$.

2 Syntaktyka języka

Sygnaturą języka nazwiemy $\Upsilon = (\mathcal{F}, \mathcal{A}_c, \mathcal{G})$, $\mathcal{F}_I \subseteq \mathcal{F}$. Dalej formułę zdefiniujemy jako $\alpha = f \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \implies \beta \mid \alpha \iff \beta$. Gdzie występują także dwie specyficzne formuły logiczne: $\top$ i $\bot$, oznaczające odpowiednio prawdę i fałsz.

2.1 Wyrażenie wartości

Język $\mathcal{MAR}$ umożliwia opis wyniku wykonania ciągu akcji.

$$\alpha \ \textbf{after} \ (A_1, G_1), ..., (A_n, G_n)$$

$$\textbf{observable} \ \alpha \ \textbf{after} \ (A_1, G_1), ..., (A_n, G_n)$$

gdzie α jest formułą logiczną, a $A_i \in \mathcal{A}_c$ oraz $G_i \subseteq \mathcal{G}$ dla $i = 1, ..., n$, odpowiednio akcjami i zbiorami agentów wykonujących dane akcje. Powyższe wyrażenia mogą być rozumiane następująco. Pierwsze z nich mówi, że warunek α jest spełniony zawsze po wykonaniu sekwencji $(A_1, G_1), ..., (A_n, G_n)$ akcji przez przypisane zbiory agentów. Drugie ze zdań mówi, że warunek α jest spełniony w przynajmniej jednej możliwej ścieżce wykonania sekwencji akcji przez przypisanych agentów. Dla $n = 0$, możemy zastosować skrócony opis:

$$\textbf{initially} \ \alpha$$

2.2 Wyrażenie efektu akcji

Wyrażenie efektu akcji, opisujące wykonanie akcji A w stanie spełniającym warunek π przez agentów ze zbioru G powoduje przejście do stanu spełniającego α opisane jest przez:

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

gdzie α, π to formuły logiczne, $A \in \mathcal{A}_c$ to akcja oraz $G \subseteq \mathcal{G}$ to zbiór agentów, którzy wykonują daną akcję. W przypadku gdy $\pi \iff \top$ wyrażenie to można skrócić do

$$A \textbf{ by } G \textbf{ causes } \alpha$$

Jeśli $\forall G \in 2^{\mathcal{G}}$ spełnione jest powyższe wyrażenie, możemy w skrócie zapisać, że

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

Natomiast gdy $\alpha \iff \bot$ możemy zapisać

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

Jeśli $\forall G \in 2^{\mathcal{G}}$ spełnione jest powyższe wyrażenie, możemy w skrócie zapisać, że

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

2.3 Wyrażenie uwolnienia fluentu

Wyrażenie uwolnienia fluentu, opisujące sytuację, w której przeprowadzenie akcji A przez agentów ze zbioru G w stanie spełniającym warunek logiczny π, może, ale też nie musi, zmienić wartości inercjalnego fluentu f, opisane jest przez:

$$A \textbf{ by } G \textbf{ releases } f \textbf{ if } \pi$$

gdzie $A \in \mathcal{A}_c$, $f \in \mathcal{F}_I$, $G \subseteq \mathcal{G}$, a π to formuła logiczna. Dla $\pi \iff \top$, można skrótowo zapisać to wyrażenie jako

$$A \textbf{ by } G \textbf{ releases } f$$

Podobnie jeśli $\forall G \in 2^{\mathcal{G}}$ spełniona jest powyższe wyrażenie, możemy w skrócie zapisać, że

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

2.4 Wyrażenie integralności

Język $\mathcal{MAR}$ podobnie jak języka $\mathcal{AR}$ pozwala na opis warunku logicznego α, który jest prawdziwy dla każdego stanu.

$$\textbf{always } \alpha$$

2.5 Wyrażenie zbioru agentów

Dodatkowo jeśli wykonanie danej akcji $A \in \mathcal{A}_c$ w stanie spełniającym warunek logiczny π jest niemożliwe gdy wykonuje ją jakikolwiek agent ze zbioru $G \subseteq \mathcal{G}$ możemy taki warunek zapisać jako.

$$A \textbf{ not by } G \textbf{ if } \pi$$

2.6 Wyrażenie zachowania fluentów

Fluent f ($f \notin \mathcal{F}_I$) jest nieinercyjny gdy nie jest on brany pod uwagę minimalizacji zmian. Wyrażenie to może być opisane przez

$$\textbf{noninertial } f$$

3 Semantyka

Strukturą języka $\mathcal{L}$ klasy $\mathcal{MAR}$ jest trójka $S = (\Sigma, \sigma_0, Res)$, gdzie $\Sigma \neq \emptyset$ to zbiór możliwych stanów, $\sigma_0 \in \Sigma$ jest stanem początkowym, $Res : \mathcal{A}_c \times \Sigma \times 2^{\mathcal{G}} \rightarrow 2^{\Sigma}$ to funkcja przejścia między stanami - Res przypisuje wszystkie stany osiągalne ze stanu σ, przez agentów G po wykonaniu akcji A. $Res(A, \sigma, G)$ to zbiór stanów możliwie najbliższych do σ. Σ jest zbiorem spełniającym wszystkie wyrażenia więzów.

Funkcja przejścia Niech $S = (\Sigma, \sigma_0, Res)$ będzie strukturą języka. Zdefiniujmy częściowe przejście $\Psi_S : \mathcal{A}_c \times \Sigma \times 2^{\mathcal{G}} \rightarrow \Sigma$ następująco:

- $\forall Q \subseteq \mathcal{G}\ \Psi_S(\epsilon, \sigma, Q) = \sigma$
- jeśli dla jakiegokolwiek wyrażenia (A **not by** Q **if** π), $Q \cap G \neq \emptyset$ to $\Psi_S(A, \sigma, G)$ jest niezdefiniowane

Uogólniona funkcja przejścia Funkcja ta może zostać uogólniona do postaci $\Psi^*_S : 2^{\mathcal{A}_c} \times \Sigma' \times 2^{2^{\mathcal{G}}} \rightarrow \Sigma'$, jako:

$$\Psi^*_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n)) = \Psi_S(A_n, \Psi_S((A_1, ..., A_{n-1}), \sigma, (G_1, ..., G_{n-1})), G_n)$$

$$\Psi^*_S(\epsilon, \sigma, Q) = \sigma, Q \subseteq \mathcal{G}$$

Jeśli $\Psi^*_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$, $n \geqslant 1$, jest zdefiniowane to

$$\Psi^*_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n)) \in Res(A_n, \Psi^*_S((A_1, ..., A_{n-1}), \sigma, (G_1, ..., G_{n-1})), G_n)$$

Dla dalszej części przyjmijmy że D jest dziedziną akcji - skończonym zbiorem stwierdzeń języka $\mathcal{MAR}$, a $S = (\Sigma, \sigma_0, Res)$ jest strukturą $\mathcal{MAR}$.

Fluent inercjalny Fluent f jest inercjalny w D wtedy i tylko wtedy gdy

$$(\textbf{intertial}\ \ f) \notin D$$

Stan dziedziny Stan $\sigma : \mathcal{F} \rightarrow \{0, 1\}$ jest stanem D wtedy i tylko wtedy gdy dla każdego wyrażenia integraloności (**always** α) $\in D$, spełniony jest warunek

$$\sigma \models \alpha$$

Spełnienie wyrażenia wartości

- Wyrażenie wartości (α **after** $(A_1, G_1), ..., (A_n, G_n)$) jest prawdziwe w strukturze S wtedy i tylko wtedy gdy
$$\forall \Psi_S\ \ \Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n)) \models \alpha$$
- Wyrażenie wartości (**observable** α **after** $(A_1, G_1), ..., (A_n, G_n)$) jest prawdziwe w strukturze S wtedy i tylko wtedy gdy
$$\exists \Psi_S\ \ \Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n)) \models \alpha$$

Pomocnicza funkcja Res_0 Zdefiniujmy pomocniczą funkcję $Res_0 : \mathcal{A}_c \times \Sigma \times 2^{\mathcal{G}} \rightarrow 2^{\Sigma}$ następująco, dla każdych $A \in \mathcal{A}_c, \sigma \in \Sigma, G \in 2^{\mathcal{G}}$

$$Res_0(A, \sigma, G) = \{\sigma' \in \Sigma : (A\ \textbf{by}\ Q\ \textbf{causes}\ \alpha\ \textbf{if}\ \pi) \in D\ \wedge\ Q \subseteq G\ \wedge\ (\sigma \models \pi) \Longrightarrow (\sigma' \models \alpha)\}$$

Funkcja New Dla każdego $\sigma \in \Sigma, A \in \mathcal{A}_c, G \in 2^{\mathcal{G}}$, oraz dla każdego $\sigma' \in Res_0(A, \sigma, G)$, $New(A, \sigma, \sigma')$ jest zbiorem literałów $\overline{f}$, takich że $\sigma' \models \overline{f}$ oraz

- $f \in \mathcal{F}_I\ \wedge\ \sigma(f) \neq \sigma'(f)$, lub
- istnieje takie stwierdzenie (A **by** Q **releases** f **if** π) $\in D\ \wedge\ \sigma \models \pi\ \wedge\ Q \subseteq G$

3.1 Model

Niech D będzie dziedziną akcji w języku $\mathcal{L}$ klasy $\mathcal{MAR}$ oraz niech $S = (\Sigma, \sigma_0, Res)$ będzie strukturą dla $\mathcal{L}$. Mówimy, że S jest modelem D wtedy i tylko wtedy gdy:

- Σ jest zbiorem wszystkich stanów D,
- każde wyrażenie wartości i obserwacji w D jest prawdziwe w S,
- dla każdego $A \in \mathcal{A}_c$, dla każdego $\sigma \in \Sigma$, oraz dla każdego $G \in 2^{\mathcal{G}}$, $Res(A, \sigma, G)$ jest zbiorem wszystkich stanów $\sigma' \in \Sigma$, dla których zbiór $New(A, \sigma, \sigma')$ jest najmniejszy względem inkluzji zbiorów.

4 Kwerendy

Kwerendą klasy języków $\mathcal{MAR}$ nazywamy dowolne wyrażenie w jednej z opisanych poniżej postaci. Dla uproszczenia notacji, ciąg $\mathbb{P} = (A_1, G_1), ..., (A_n, G_n)$, gdzie A_i jest akcją, zaś G_i jest grupą agentów, nazywać będziemy programem działań.

4.1 Kwerenda wykonywalności

necessary executable $\mathbb{P}$ **from** π

possibly executable $\mathbb{P}$ **from** π

Zgodnie z intuicją pierwsza kwerenda mówi o tym, że program działań $\mathbb{P}$ jest **zawsze** możliwy do realizacji z dowolnego stanu spełniającego warunek π.

Druga kwerenda mówi o tym, że program działań $\mathbb{P}$ jest **czasami** możliwy do realizacji z dowolnego stanu spełniającego warunek π.

Jeśli fraza '**from** π' jest pominięta, to kwerenda odwołuje się do wykonywalności sekwencji akcji ze stanu początkowego.

4.2 Kwerendy wartości

necessary α **after** $\mathbb{P}$ **from** π

possibly α **after** $\mathbb{P}$ **from** π

Pierwsza kwerenda mówi o tym, że warunek logiczny α jest spełniony **zawsze** po wykonaniu określonej programu $\mathbb{P}$ z dowolnego stanu spełniającego warunek π.

Druga kwerenda mówi o tym, że warunek logiczny α **czasami** jest spełniony po wykonaniu programu $\mathbb{P}$ z dowolnego stanu spełniającego warunek π.

Jeśli ograniczenie '**from** π' jest pominięte, to kwerenda odwołuje się do stanu początkowego.

4.3 Kwerenda zaangażowania

necessary G **engaged in** $A_1, ..., A_n$ **from** π

possibly G **engaged in** $A_1, ..., A_n$ **from** π

Pierwsza kwerenda mówi o tym, że grupa agentów G jest zawsze wymagana do ukończenia sekwencji akcji $A_1, ..., A_n$ ze stanu spełniającego π.

Druga kwerenda mówi o tym, że grupa agentów G jest czasami zaangażowana w wykonywanie sekwencji akcji $A_1, ..., A_n$ ze stanu spełniającego π.

Jeśli ograniczenie '**from** π' jest pominięte, to kwerenda odwołuje się do stanu początkowego.

5 Spełnialność kwerend

Niech D będzie dziedziną akcji oraz niech Q będzie zapytaniem w klasie języków $\mathcal{MAR}$. Mówimy, że Q jest **konsekwencją** D (oznaczane jako $D \mid\approx Q$), wtedy i tylko wtedy, gdy:

- jeśli Q jest kwerendą wartości w postaci

  necessary executable $\mathbb{P}$ **from** π

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu $\sigma \in \Sigma$, oraz dla **każdej** funkcji przejścia Ψ_S, jeśli $\sigma \models \pi$ to $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$ jest określone.

- jeśli Q jest kwerendą wartości w postaci

  necessary executable $\mathbb{P}$

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla **każdej** funkcji przejścia Ψ_S, prawdą jest, że $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n))$ jest określone.

- jeśli Q jest kwerendą wartości w postaci

  possibly executable $\mathbb{P}$ **from** π

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu $\sigma \in \Sigma$, jeśli $\sigma \models \pi$ to **istnieje** funkcja przejścia Ψ_S taka, że $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$ jest określone.

- jeśli Q jest kwerendą wartości w postaci

  possibly executable $\mathbb{P}$

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, **istnieje** funkcja przejścia Ψ_S, taka, że $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n))$ jest określone.

- jeśli Q jest kwerendą wartości w postaci

  necessary α **after** $\mathbb{P}$ **from** π

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu $\sigma \in \Sigma$, oraz dla **każdej** funkcji przejścia Ψ_S, $\sigma \models \pi$ i $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$ jest określone implikuje $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n)) \models \alpha$

- jeśli Q jest kwerendą wartości w postaci

  necessary α **after** $\mathbb{P}$

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla **każdej** funkcji przejścia Ψ_S, jeśli $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n))$ jest określone to
  $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n)) \models \alpha$.

- jeśli Q jest kwerendą wartości w postaci

  possibly α **after** $\mathbb{P}$ **from** π

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu $\sigma \in \Sigma$, jeśli Ψ_S, $\sigma \models \pi$, to **istnieje** funkcja przejścia $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n)) \models \alpha$

- jeśli Q jest kwerendą wartości w postaci

  possibly α **after** $\mathbb{P}$

  $D \mid\approx Q$ wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, **istnieje** funkcja przejścia Ψ_S, taka, że $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n)) \models \alpha$.

- jeśli Q jest kwerendą wartości w postaci

$$\textbf{necessary } G \textbf{ engaged in } A_1, ..., A_n \textbf{ from } \pi$$

D $|\approx$ Q wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu dla każdego stanu $\sigma \in \Sigma$, oraz dla **każdej** funkcji przejścia Ψ_S, $\sigma \models \pi$ implikuje, fakt, że $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$ jest określone implikuje $\Psi_S((A_1, ..., A_n), \sigma, ((G_1 \setminus G), ..., (G_n \setminus G))$ nie jest określone.

- jeśli Q jest kwerendą wartości w postaci **??**

$$\textbf{necessary } G \textbf{ engaged in } A_1, ..., A_n$$

D $|\approx$ Q wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla **każdej** funkcji przejścia Ψ_S, fakt, że $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n))$ jest określone implikuje, że $\Psi_S((A_1, ..., A_n), \sigma_0, ((G_1 \setminus G), ..., (G_n \setminus G))$ nie jest określone.

- jeśli Q jest kwerendą wartości w postaci

$$\textbf{possibly } G \textbf{ engaged in } A_1, ..., A_n \textbf{ from } \pi$$

D $|\approx$ Q wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, dla każdego stanu dla każdego stanu $\sigma \in \Sigma$, jeśli $\sigma \models \pi$ to **istnieje** funkcja przejścia $\Psi_S((A_1, ..., A_n), \sigma, (G_1, ..., G_n))$ i $\Psi_S((A_1, ..., A_n), \sigma, ((G_1 \setminus G), ..., (G_n \setminus G))$ nie jest określone.

- jeśli Q jest kwerendą wartości w postaci

$$\textbf{possibly } G \textbf{ engaged in } A_1, ..., A_n$$

D $|\approx$ Q wtedy i tylko wtedy gdy dla każdego modelu $S = (\Sigma, \sigma_0, \mathcal{G}, Res)$ dziedziny D, **istnieje** funkcja przejścia Ψ_S, taka, że $\Psi_S((A_1, ..., A_n), \sigma_0, (G_1, ..., G_n))$ jest określone i $\Psi_S((A_1, ..., A_n), \sigma_0, ((G_1 \setminus G), ..., (G_n \setminus G))$ nie jest określone.

6 Przykłady

6.1 Przykład 1.

Samochód wiozący A, B, C i D zepsuł się w trasie. Jedyną szansą na kontynuację jazdy jest uruchomienie go poprzez popchnięcie. Samochód jest na tyle ciężki, że jedna osoba nie jest w stanie na tyle go rozpędzić, aby uruchomić silnik. B i C nie są na tyle wysportowani, aby uruchomić samochód. A jest silnym mężczyzną, ale sam też sobie nie poradzi. D śpi i koledzy nie chcą go budzić. Jeden z pasażerów musi siedzieć za kierownicą aby uruchomić samochód.

initially ¬isRunning

impossible PUSH **if** *isRunning*

impossible PUSH **by** {D}

PUSH **by** {A} **causes** ¬*isRunning*

PUSH **by** {B} **causes** ¬*isRunning*

PUSH **by** {C} **causes** ¬*isRunning*

PUSH **by** {B,C} **causes** ¬*isRunning*

PUSH **by** {A,B} **causes** *isRunning*

PUSH **by** {A,C} **causes** *isRunning*

PUSH **by** {A,B,C} **causes** ¬*isRunning*

$\sigma_0 = \{\neg isRunning\}$

$\sigma_1 = \{isRunning\}$

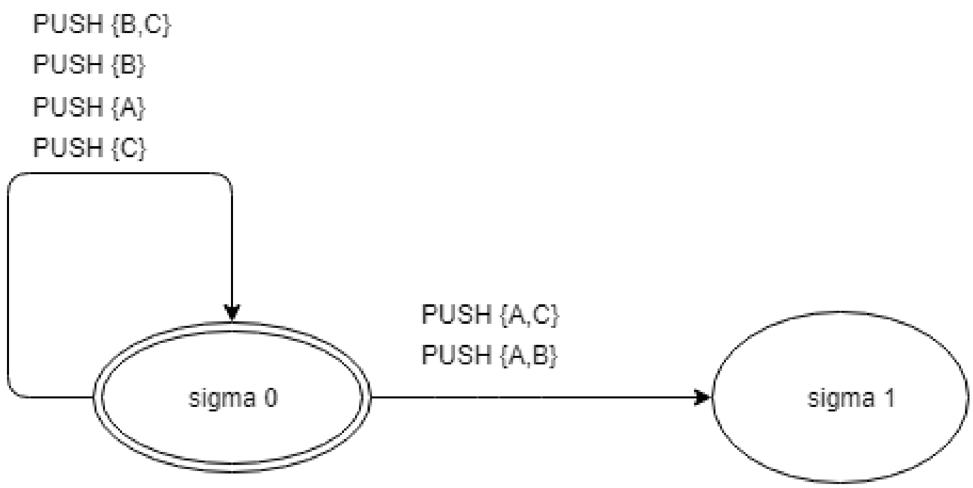
$$Res_0(PUSH, \sigma_0, \{A\}) = \{\sigma_0\}$$
$$New(PUSH, \sigma_0, \sigma_0) = \emptyset$$
$$Res(PUSH, \sigma_0, \{A\}) = \{\sigma_0\}$$

Dla zbiorów agentów {B}, {C} i {B,C} obliczenia są identyczne z obliczeniami dla zbioru {A}.

$$Res_0(PUSH, \sigma_0, \{A, B\}) = \{\sigma_1\}$$
$$New(PUSH, \sigma_0, \sigma_1) = \{isRunning\}$$
$$Res(PUSH, \sigma_0, \{A, B\}) = \{\sigma_1\}$$

Dla zbiorów agentów {A,C} obliczenia są identyczne z obliczeniami dla zbioru {A,B}.

$$Res_0(PUSH, \sigma_0, \{A, B, C\}) = \{\sigma_0\}$$
$$New(PUSH, \sigma_0, \sigma_0) = \emptyset$$
$$Res(PUSH, \sigma_0, \{A, B, C\}) = \{\sigma_0\}$$



6.1.1 Kwerendy

neccessary executable (PUSH,{A,B}) from $\neg isRunning$
neccessary executable (PUSH,{A,C}) from $\neg isRunning$

neccessary isRunning **after** (PUSH,{A,B}) from $\neg isRunning$
neccessary isRunning **after** (PUSH,{A,C}) from $\neg isRunning$

neccessary {A} engaged in PUSH
possibly {B} engaged in PUSH
possibly {C} engaged in PUSH

6.2 Przykład 2.

Robotnicy A i B wciągają ciężki ładunek na linach na pewną wysokość. Ciężar jest na tyle duży, że do jego wciągnięcia potrzeba obu pracowników. Na górze ładunek można zabezpieczyć. Jeśli ładunek jest na górze i niezabezpieczony, obu pracowników musi go tam utrzymać. Praca tylko jednego z nich kończy się upadkiem ładunku i śmiercią obu pracowników. Ciężar można zabezpieczyć tylko jeśli jest na górze.

initially $alive$

initially $\neg protected$

initially $\neg up$

LIFT by {A,B} causes up

LIFT by {A} causes $\neg alive$ if $\neg protected \wedge up$

LIFT by {B} causes $\neg alive$ if $\neg protected \wedge up$

PROTECT causes $protected$ if up

impossible LIFT if $protected \vee \neg alive$

impossible PROTECT if $protected \vee \neg alive \vee \neg up$

$\sigma_0 = \{\neg up, \neg protected, alive\}$

$\sigma_1 = \{up, \neg protected, alive\}$

$\sigma_2 = \{up, protected, alive\}$

$\sigma_3 = \{\neg up, \neg protected, \neg alive\}$

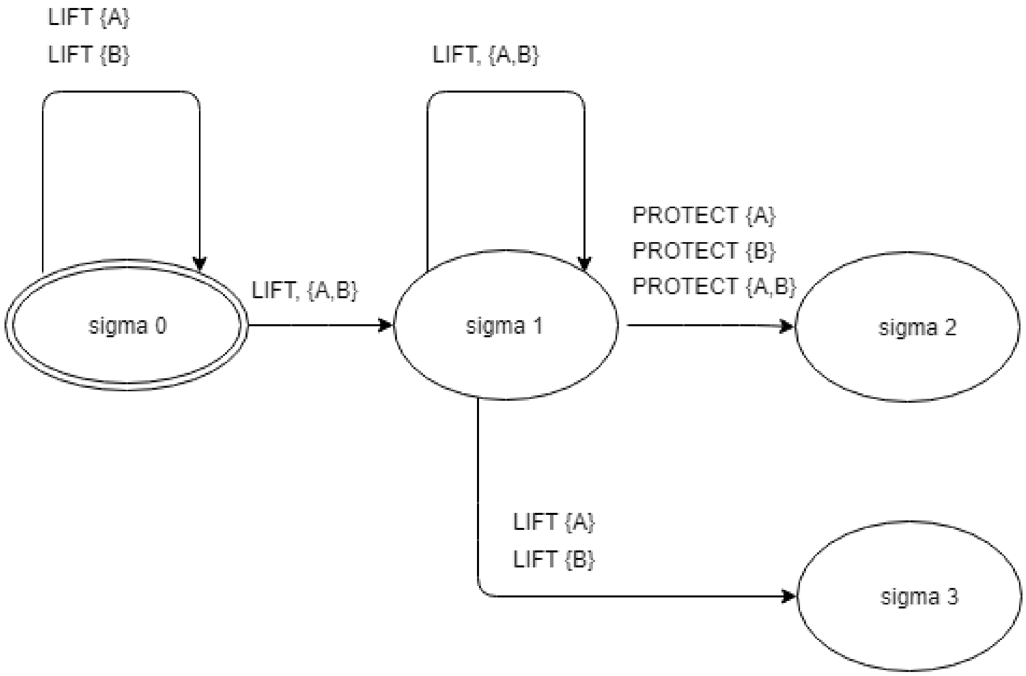
$$Res_0(LIFT, \sigma_0, \{A, B\}) = \{\sigma_1, \sigma_2\}$$
$$New(LIFT, \sigma_0, \sigma_1) = \{up\}$$
$$New(LIFT, \sigma_0, \sigma_2) = \{up, protected\}$$
$$Res(LIFT, \sigma_0, \{A, B\}) = \{\sigma_1\}$$

$$Res_0(LIFT, \sigma_0, \{A\}) = Res_0(LIFT, \sigma_0, \{B\}) = \{\sigma_0\}$$
$$Res(LIFT, \sigma_0, \{A\}) = Res(LIFT, \sigma_0, \{B\}) = \{\sigma_0\}$$

$$Res_0(LIFT, \sigma_1, \{A, B\}) = \emptyset$$
$$Res(LIFT, \sigma_1, \{A, B\}) = \{\sigma_1\}$$

$$Res_0(LIFT, \sigma_1, \{A\}) = Res_0(LIFT, \sigma_1, \{B\}) = \{\sigma_3\}$$
$$Res(LIFT, \sigma_1, \{A\}) = Res(LIFT, \sigma_1, \{B\}) = \{\sigma_3\}$$

$$Res_0(PROTECT, \sigma_1, Q) = \{\sigma_2\}$$
$$Res(PROTECT, \sigma_1, Q) = \{\sigma_2\}$$



6.2.1 Kwerendy

necessary executable (LIFT, {A,B}) from *alive*

necessary executable (PROTECT,{A}) from $up \wedge alive$

necessary executable (PROTECT,{B}) from $up \wedge alive$

necessary up after (LIFT,{A,B}) from *alive*

necessary protected after (PROTECT,{A}) from *up*

necessary protected after (PROTECT,{B}) from *up*

possibly {A} engaged in PROTECT from *up*

possibly {B} engaged in PROTECT from *up*

6.3 Przykład 3.

Bercik i Filemon to dwaj początkujący alchemicy. Dostali za zadanie stworzyć magiczną miksturę, jednak woleliby leżeć na łóżku i nic nie robić. Mieszkają w nowo wyremontowanej drewnianej chatce. Uzupełniają swoje wady, więc jeśli zabiorą się do warzenia mikstury wspólnie, uda im się. Jednakże, jeśli zabierze się za to tylko jeden z nich, mikstura może nie wyjść i będzie trzeba tworzyć ją ponownie. Dodatkowo, jeśli miksturę warzyć będzie sam Bercik, może się zdarzyć, że ta wybuchnie i zniszczy nowiutko wyremontowaną chatkę, przez co nie stworzą już mikstury. Z uwagi na lenistwo alchemików, jeśli uda im się uwarzyć miksturę, na pewno nie będzie im się chciało tworzyć kolejnej.

initially $\neg brewed$

initially $\neg destroyed$

impossible BREW **if** $brewed \vee destroyed$

BREW **by** {Bercik, Filemon} **causes** *brewed*

BREW **by** {Bercik} **releases** *brewed*

BREW **by** {Bercik} **releases** *destroyed*

BREW **by** {Filemon} **releases** *brewed*

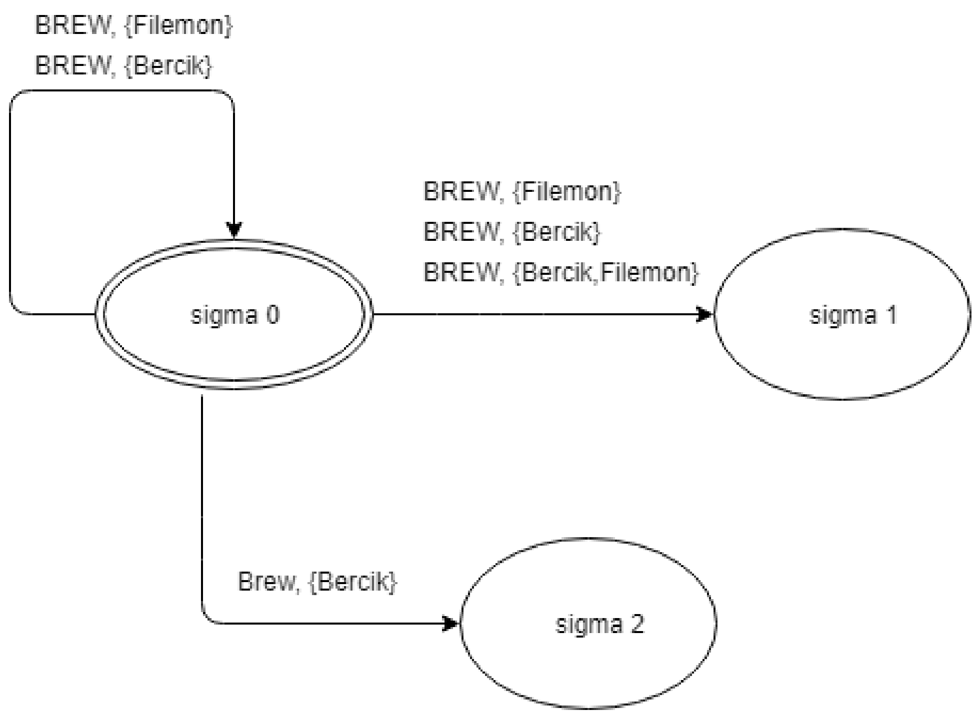
$\sigma_0 = \{\neg brewed, \neg destroyed\}$

$\sigma_1 = \{brewed, \neg destroyed\}$
$\sigma_2 = \{\neg brewed, destroyed\}$

$$Res_0(BREW, \sigma_0, \{Bercik, Filemon\}) = \{\sigma_1\}$$
$$New(BREW, \sigma_0, \sigma_1) = \{brewed\}$$
$$Res(BREW, \sigma_0, \{A\}) = \{\sigma_1\}$$

$$Res_0(BREW, \sigma_0, \{Bercik\}) = \{\sigma_0, \sigma_1, \sigma_2\}$$
$$New(BREW, \sigma_0, \sigma_0) = \{\emptyset\}$$
$$New(BREW, \sigma_0, \sigma_1) = \{brewed\}$$
$$New(BREW, \sigma_0, \sigma_2) = \{destroyed\}$$
$$Res(BREW, \sigma_0, \{Bercik\}) = \{\sigma_0, \sigma_1, \sigma_2\}$$

$$Res_0(BREW, \sigma_0, \{Filemon\}) = \{\sigma_0, \sigma_1\}$$
$$New(BREW, \sigma_0, \sigma_0) = \{\emptyset\}$$
$$New(BREW, \sigma_0, \sigma_1) = \{brewed\}$$
$$Res(BREW, \sigma_0, \{Bercik\}) = \{\sigma_0, \sigma_1\}$$



6.3.1 Kwerendy

neccessary executable (BREW) from $\neg brewed \wedge \neg destroyed$
possibly executable (BREW) from $\neg brewed$
possibly executable (BREW) from $\neg destroyed$

neccessary brewed **after** (BREW,{Bercik, Filemon}) from $\neg brewed \wedge \neg destroyed$
possibly brewed **after** (Brew,{Filemon}) from $\neg brewed \wedge \neg destroyed$

possibly brewed **after** (Brew,{Bercik}) from $\neg brewed \wedge \neg destroyed$

possibly brewed **after** (Brew,{Bercik}),(Brew,{Bercik}),(Brew,{Filemon}) from $\neg brewed \wedge \neg destroyed$

possibly destroyed **after** (Brew,{Bercik}) from $\neg brewed \wedge \neg destroyed$

possibly {Bercik} engaged in BREW

possibly {Filemon} engaged in BREW