



Wydział Matematyki i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Programy działań z efektami domyślnymi

Reprezentacja Wiedzy - projekt

Bartosz Borkowicz
Sebastian Konowrocki
Tomasz Laskowski
Piotr Łazarczyk
Jakub Niemyjski
Bartłomiej Pieńkowski
Igor Pieńkowski
Bartłomiej Teodorczuk
Łukasz Wasilewski

Wersja 1.0

28 maja 2018

Spis treści

1	Opis zadania	2
1.1	Oznaczenia	2
2	Język akcji	3
2.1	Syntaktyka	3
2.1.1	Formuły	3
2.1.2	Instrukcja wartości	4
2.1.3	Instrukcja efektu	4
2.1.4	Instrukcja uwolnienia	5
2.1.5	Instrukcja ograniczenia	5
2.1.6	Instrukcja opisu fluentu	5
2.2	Semantyka	6
3	Język kwerend	8
3.1	Syntaktyka	8
3.2	Semantyka	8
4	Przykład	10
4.1	Kwerendy	12

1 Opis zadania

Celem projektu jest opracowanie języka akcji oraz odpowiadającego mu języka kwerend dla klasy systemów dynamicznych, która została zdefiniowana poniżej.

Język akcji ma być zaprojektowany dla klasy systemów dynamicznych spełniających następujące warunki:

1. Prawo inercji
2. Niedeterminizm i sekwencyjność działań
3. Pełna informacja o wszystkich akcjach i wszystkich skutkach bezpośrednich
4. Z każdą akcją związany jest
 - warunek początkowy (ew. *true*)
 - efekt akcji
 - jej wykonawca
5. Skutki akcji:
 - *pewne* - zawsze występują po zakończeniu akcji
 - *domyślne* - preferowane, zachodzą po zakończeniu akcji, o ile nie jest wiadomym, że nie występują
6. Efekty akcji zależą od stanu, w którym akcja się zaczyna oraz od jej wykonawcy
7. W pewnych stanach akcje mogą być niewykonalne przez pewnych (wszystkich) wykonawców

Język akcji spełniający dane warunki może być odpytywany poprzez odpowiadający mu język kwerend, który zapewnia uzyskanie odpowiedzi prawda (*true*) bądź fałsz (*false*) na następujące pytania:

1. Czy podany program działań jest wykonywalny zawsze/kiedykolwiek?
2. Czy wykonanie podanego programu działań z dowolnego stanu spełniającego warunek π prowadzi zawsze/kiedykolwiek/na ogólny do stanu spełniającego warunek celu γ ?
3. Czy z dowolnego stanu spełniającego warunek π cel γ jest osiągany zawsze/kiedykolwiek/na ogólny?
4. Czy wskazany wykonawca jest zaangażowany w realizację programu zawsze/kiedykolwiek?

1.1 Oznaczenia

Na potrzeby uproszczenia definicji oraz opisów w dokumencie wykorzystywane są następujące oznaczenia:

- $A \in \mathcal{A}c$ - akcje
- $\omega \in \Omega$ - agenci, w szczególności ε oznacza brak agenta (jakiegokolwiek agenta)
- $f \in \mathcal{F}$ - fluenty
- \bar{f} - literał odpowiadający f , czyli f lub $\neg f$
- $\alpha, \pi \in \text{Forms}(\mathcal{F})$ - formuły
- $\sigma \in \Sigma$ - stany

Formułą nazywamy dowolne połączenie fluentów:

$$\text{Forms}(\mathcal{F}) \ni \alpha = f \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \Rightarrow \alpha_2 \mid \alpha_1 \Leftrightarrow \alpha_2$$

Dodatkowo istnieją specjalne formuły:

- \top oznaczający zawsze prawdę (*true*)
- \perp oznaczający zawsze fałsz (*false*)

2 Język akcji

Definicja 2.1 (Język ADA) Język ADA jest językiem spełniającym założenia podane w opisie zadania. W poniższych sekcjach zostanie zdefiniowana jego sygnatura, syntaktyka oraz semantyka.

Definicja 2.2 (Sygnatura) Sygnaturą języka ADA nazywamy parę $\Upsilon = (\mathcal{F}, \mathcal{Ac})$, gdzie:

- \mathcal{F} jest zbiorem fluentów
- \mathcal{Ac} jest zbiorem akcji

2.1 Syntaktyka

Identyfikatorem jest nazwa akcji, agenta lub fluentu. Identyfikator składa się z ciągu dużych i małych liter alfabetu łacińskiego, cyfr oraz znaku podkreślenia ($\underline{}$). Dodatkowo identyfikator nie może być żadnym ze słów kluczowych: **initially**, **always**, **observable**, **after**, **when**, **causes**, **impossible**, **typically**, **noninertial**, **if**, **not**, **and**, **or**, **then**, **iff**, **true**, **false**. W przypadku identyfikatorów wielkość liter ma znaczenie, to znaczy przykładowo akcje *shoot* i *Shoot* są dwiema różnymi akcjami.

2.1.1 Formuły

Poszczególnym operacjom z konstrukcji formuł odpowiadają następujące zapisy:

- $f \rightarrow \text{identyfikator}$ - nazwa fluentu, będąca identyfikatorem zgodnym z opisem powyżej,
- $\neg\alpha \rightarrow \text{not } \alpha$ - negacja logiczna formuły α ,
- $\alpha_1 \wedge \alpha_2 \rightarrow \alpha_1 \text{ and } \alpha_2$ - koniunkcja logiczna formuł α_1 i α_2 ,
- $\alpha_1 \vee \alpha_2 \rightarrow \alpha_1 \text{ or } \alpha_2$ - alternatywa logiczna formuł α_1 i α_2 ,
- $\alpha_1 \Rightarrow \alpha_2 \rightarrow \alpha_1 \text{ then } \alpha_2$ - implikacja logiczna formuły α_1 w α_2 ,
- $\alpha_1 \Leftrightarrow \alpha_2 \rightarrow \alpha_1 \text{ iff } \alpha_2$ - równoważność logiczna (*if and only if*) formuł α_1 i α_2 ,
- $\top \rightarrow \text{true}$ - oznacza zawsze prawdę,
- $\perp \rightarrow \text{false}$ - oznacza zawsze fałsz.

Dla wyżej wymienionych operacji została zdefiniowana następująca kolejność działań:

1. **not**
2. **and**, **or**
3. **then**
4. **iff**

Nawiasy okrągłe () mogą być użyte w celu grupowania wyrażeń oraz zmiany kolejności wykonywania działań matematycznych.

2.1.2 Instrukcja wartości

$\alpha \text{ after } \omega_1 A_1, \dots, \omega_n A_n$
gdzie $n \geq 1$

Formuła α jest **zawsze** prawdziwa po wykonaniu sekwencji akcji A_1 przez agenta ω_1, \dots, A_n przez agenta ω_n . Podanie agenta wykonującego akcję nie jest wymagane. Jeżeli agent nie zostanie podany, wtedy akcja jest wykonywana przez kogokolwiek.

initially α

Jeżeli dla powyższej instrukcji $n = 0$, to musimy użyć skróconego zapisu. Oznacza on, że formuła α jest spełniona dla stanu początkowego.

observable $\alpha \text{ after } \omega_1 A_1, \dots, \omega_n A_n$
gdzie $n \geq 1$

Formuła α jest **czasem** prawdziwa po wykonaniu sekwencji akcji A_1 przez agenta ω_1, \dots, A_n przez agenta ω_n . Podanie agenta wykonującego akcję nie jest wymagane. Jeżeli agent nie zostanie podany, wtedy akcja jest wykonywana przez kogokolwiek.

2.1.3 Instrukcja efektu

when $\omega_1 \text{ or } \dots \text{ or } \omega_n A$ **causes** α **if** π (*)
gdzie $n \geq 1$

Jeżeli akcja A jest wywołana przez kogokolwiek z agentów $\omega_1, \dots, \omega_n$ w stanie, który spełnia formułę π , stan wynikowy spełnia formułę α .

when $\omega_1 \text{ or } \dots \text{ or } \omega_n A$ **causes** α
gdzie $n \geq 1$

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Jeżeli akcja A jest wywołana przez kogokolwiek z agentów $\omega_1, \dots, \omega_n$, stan wynikowy spełnia formułę α .

A **causes** α **if** π

Jeżeli akcja A jest wywołana przez dowolnego agenta w stanie, który spełnia formułę π , stan wynikowy spełnia formułę α . Jest to zapis instrukcji (*), jeżeli akcja jest wywoływana przez dowolnego agenta.

A **causes** α

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Jeżeli akcja A jest wywołana, stan wynikowy spełnia formułę α .

impossible $\omega_1 \text{ or } \dots \text{ or } \omega_n A$ **if** π
gdzie $n \geq 1$

Zapis równoważny z (*), jeżeli $\alpha \equiv \perp$. Akcja A nie jest wykonywalna przez żadnego z agentów $\omega_1, \dots, \omega_n$, w stanie, który spełnia formułę α .

impossible $\omega_1 \text{ or } \dots \text{ or } \omega_n A$
gdzie $n \geq 1$

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Akcja A nie jest wykonywalna przez żadnego z agentów $\omega_1, \dots, \omega_n$.

impossible A **if** π

Akcja A nie jest wykonywalna przez żadnego agenta w stanie, który spełnia formułę α .

when ω_1 **or** ... **or** ω_n A **typically causes** α **if** π
 gdzie $n \geq 1$

Jeżeli akcja A jest wywołana przez którykolwiek z agentów $\omega_1, \dots, \omega_n$ w stanie, który spełnia formułę π , stan wynikowy z reguły powinien spełniać formułę α . Efekt α jest **preferowany** w odróżnieniu do efektów niedeterministycznych, gdzie żaden wynik nie jest preferowany.

when ω_1 **or** ... **or** ω_n A **typically causes** α
 gdzie $n \geq 1$

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Jeżeli akcja A jest wywołana przez którykolwiek z agentów $\omega_1, \dots, \omega_n$, stan wynikowy z reguły powinien spełniać formułę α .

A typically causes α if π

Jeżeli akcja A jest wywołana przez dowolnego agenta w stanie, który spełnia formułę π , stan wynikowy powinien spełniać formułę α .

A typically causes α

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Jeżeli akcja A jest wywołana, stan wynikowy powinien spełniać formułę α .

2.1.4 Instrukcja uwolnienia

when ω_1 **or** ... **or** ω_n A **releases** \bar{f} **if** π (**)

Wykonanie akcji A przez którykolwiek agentów $\omega_1, \dots, \omega_n$ w stanie, który spełnia warunek π **może**, ale **nie musi** zmienić wartość inercyjnego fluentu f .

when ω_1 **or** ... **or** ω_n A **releases** \bar{f}

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Wykonanie akcji A którykolwiek agentów $\omega_1, \dots, \omega_n$ **może**, ale **nie musi** zmienić wartość inercyjnego fluentu f .

A releases \bar{f} if π

Wykonanie akcji A przez dowolnego agenta w stanie, który spełnia warunek π **może**, ale **nie musi** zmienić wartość inercyjnego fluentu f . Jest to zapis instrukcji (**), jeżeli akcja jest wywoływana przez dowolnego agenta.

A releases \bar{f}

Zapis równoważny z powyższym, jeżeli $\pi \equiv \top$. Wykonanie akcji A **może**, ale **nie musi** zmienić wartość inercyjnego fluentu f .

2.1.5 Instrukcja ograniczenia

always α

Każdy stan musi spełniać formułę α .

2.1.6 Instrukcja opisu fluentu

noninertial f

Oznaczenia fluentu f jako nieinercjalnego.

2.2 Semantyka

Definicja 2.3 (Stan) Stanem nazywamy funkcję $\sigma : \mathcal{F} \rightarrow \{0, 1\}$. Jeżeli $f \in \mathcal{F}$ i $\sigma(f) = 1$ to zapisujemy $\sigma \models f$. Jeżeli $\alpha \in \text{Forms}(\mathcal{F})$ i formula α jest prawdziwa w stanie σ (σ jest modelem α), to zapisujemy $\sigma \models \alpha$.

Definicja 2.4 (Dziedzina akcji) Dziedziną akcji D nazywamy dowolny niepusty zbiór instrukcji.

Definicja 2.5 (Semantyczna struktura interpretacyjna języka \mathcal{ADA}) Semantyczną strukturą interpretacyjną języka \mathcal{ADA} nazywamy czwórkę $S = (\Sigma, \sigma_0, \text{ResAb}, \text{ResN})$, gdzie:

- Σ jest zbiorem stanów
- $\sigma_0 \in \Sigma$ jest stanem początkowym
- $\text{ResAb} : \mathcal{Ac} \times \Omega \times \Sigma \rightarrow 2^\Sigma$ jest nietypową funkcją przejścia
- $\text{ResN} : \mathcal{Ac} \times \Omega \times \Sigma \rightarrow 2^\Sigma$ jest typową funkcją przejścia

Niech D będzie dziedziną akcji języka \mathcal{ADA} i niech $S = (\Sigma, \sigma_0, \text{ResAb}, \text{ResN})$ będzie jego strukturą. Stan σ jest stanem dziedziny D wtedy i tylko wtedy, gdy dla każdej instrukcji ograniczenia typu **always** α zachodzi $\sigma \models \alpha$ w danej dziedzinie D .

Definicja 2.6 (Program działań) Programem działań nazywamy ciąg $((A_1, \omega_1), \dots, (A_n, \omega_n))$, gdzie A_i jest akcją, zaś ω_i jej wykonawcą lub ε (ktokolwiek).

Definicja 2.7 (Ścieżka wykonalności) Niech $\text{Res} = \text{ResAb} \cup \text{ResN}$, ścieżką wykonalności będziemy nazywać mapowanie $\Psi_S : (\mathcal{Ac} \times (\Omega \cup \{\varepsilon\}))^* \times \Sigma \rightarrow 2^\Sigma$ takie, że:

- $\Psi_S(\varepsilon, \sigma) = \sigma$
- jeśli $\Psi_s(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma)$ jest zdefiniowany, to

$$\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma) \in \begin{cases} \bigcup_{\omega \in \Omega} \text{Res}(A_n, \omega, \Psi_S((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1}))), \text{ jeśli } \omega_n = \varepsilon \\ \text{Res}(A_n, \omega_n, \Psi_S((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1}))), \text{ w.p.p.} \end{cases}$$

Instrukcja wartości typu (**observable**) α after $\omega_1 A_1, \dots, \omega_n A_n$ jest prawdziwa w S wtedy i tylko wtedy, gdy dla (jakiegokolwiek) **każdego** mapowania Ψ_S

- $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma)$ jest zdefiniowane oraz
- $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma) \models \alpha$

Dodatkowo zdefiniujmy ścieżki wykonalności pokrywające pojedyncze klasy funkcji przejścia:

Atypową ścieżką wykonalności będziemy nazywać mapowanie $\Psi_{SAb} : (\mathcal{Ac} \times (\Omega \cup \{\varepsilon\}))^* \times \Sigma \rightarrow 2^\Sigma$ takie, że:

- $\Psi_{SAb}(\varepsilon, \sigma) = \sigma$
- jeśli $\Psi_s(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma)$ jest zdefiniowany, to

$$\Psi_{SAb}(((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma) \in \begin{cases} \bigcup_{\omega \in \Omega} \text{ResAb}(A_n, \omega, \Psi_{SAb}((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1}))), \text{ jeśli } \omega_n = \varepsilon \\ \text{ResAb}(A_n, \omega_n, \Psi_{SAb}((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1}))), \text{ w.p.p.} \end{cases}$$

Typową ścieżką wykonalności będziemy nazywać mapowanie $\Psi_{SN} : (\mathcal{Ac} \times (\Omega \cup \{\varepsilon\}))^* \times \Sigma \rightarrow 2^\Sigma$ takie, że:

- $\Psi_{SN}(\varepsilon, \sigma) = \sigma$

- jeśli $\Psi_s((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma)$ jest zdefiniowany, to

$$\Psi_{SN}((A_1, \omega_1), \dots, (A_n, \omega_n), \sigma) \in \begin{cases} \bigcup_{\omega \in \Omega} ResN(A_n, \omega, \Psi_{SN}((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1})), \text{ jeśli } \omega_n = \varepsilon \\ ResN(A_n, \omega_n, \Psi_{SN}((A_1, \omega_1), \dots, (A_{n-1}, \omega_{n-1}))), \text{ w.p.p.} \end{cases}$$

Wprowadźmy następujące oznaczenia:

- $New : \mathcal{Ac} \times \Omega \times \Sigma \times \Sigma \rightarrow 2^{\bar{\mathcal{F}}}$ jest funkcją, która przypisuje każdej akcji $A \in \mathcal{Ac}$, każdemu agentowi $\omega \in \Omega$ i wszystkim $\sigma, \sigma' \in \Sigma$ zbiór literałów \bar{f} takich, że:
 - $\sigma' \models \bar{f}$
 - f jest inercjalny oraz $\sigma \models f \neq \sigma' \models f$ lub dla jakiejś instrukcji prawdziwe jest
(when ωA releases f if π) $\in D \vee (A \text{ releases } f \text{ if } \pi) \in D \wedge \sigma \models \pi$
- $Res_0 : \mathcal{Ac} \times \Omega \times \Sigma \rightarrow 2^\Sigma$ jest funkcją, która dla każdej akcji $A \in \mathcal{Ac}$, każdego stanu $\sigma \in \Sigma$ i każdego agenta $\omega \in \Omega$ jest zbiorem wszystkich stanów $\sigma' \in \Sigma$ takich, że:
(when ωA causes α if π) $\in D \vee (A \text{ causes } \alpha \text{ if } \pi) \in D \wedge \sigma \models \pi \Rightarrow \sigma' \models \alpha$
- $Res^- : \mathcal{Ac} \times \Omega \times \Sigma \rightarrow 2^\Sigma$ jest funkcją, która dla każdej akcji $A \in \mathcal{Ac}$, każdego stanu $\sigma \in \Sigma$ i każdego agenta $\omega \in \Omega$ jest zbiorem wszystkich stanów $\sigma' \in Res_0(A, \omega, \sigma)$ takich, że zbiory $New(A, \omega, \sigma, \sigma')$ są minimalne.
- $Res_0^+ : \mathcal{Ac} \times \Omega \times \Sigma \rightarrow 2^\Sigma$ jest funkcją, która dla każdej akcji $A \in \mathcal{Ac}$, każdego stanu $\sigma \in \Sigma$ i każdego agenta $\omega \in \Omega$ jest zbiorem wszystkich stanów $\sigma' \in Res_0(A, \omega, \sigma)$ takich, że:
(when ωA typically causes α if π) $\in D \vee (A \text{ typically causes } \alpha \text{ if } \pi) \in D \wedge \sigma \models \pi \Rightarrow \sigma' \models \alpha$

Definicja 2.8 (Model) Niech D będzie dziedziną akcji języka ADA. Struktura $S = (\Sigma, \sigma_0, ResAb, ResN)$ jest modelem D wtedy i tylko wtedy, gdy:

- Σ jest zbiorem wszystkich stanów dziedziny D
- wszystkie instrukcje wartości w D są prawdziwe w S
- dla każdej akcji $A \in \mathcal{Ac}$, każdego stanu $\sigma \in \Sigma$ i każdego agenta $\omega \in \Omega$
 $ResN(A, \omega, \sigma) = \{\sigma' \in Res_0^+(A, \omega, \sigma) : New(A, \omega, \sigma, \sigma') \text{ jest minimalny}\}$
 $ResAb(A, \omega, \sigma) = Res^-(A, \omega, \sigma) \setminus ResN(A, \omega, \sigma)$

Definicja 2.9 (Miara nietypowości) Dla każdej akcji $A \in \mathcal{Ac}$, dowolnego wykonawcy ω i dowolnych dwóch stanów $\sigma, \sigma' \in \Sigma$ definiujemy funkcję, zwaną miarą nietypowości:

$$\kappa_S(A, \omega, \sigma, \sigma') = \begin{cases} 0, \text{ jeśli } \sigma' \in ResN(A, \omega, \sigma) \\ 1 \text{ jeśli } \sigma' \in ResAb(A, \omega, \sigma) \\ +\infty \text{ w.p.p.} \end{cases}$$

Definicja 2.10 (Zbiór Π_S) Zdefiniujmy funkcję K_Ψ , taką że dla każdego mapowania Ψ_S oraz dowolnego programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$:

- $K_\Psi((A_1, \omega_1), \dots, (A_n, \omega_n)) = +\infty$, jeżeli $\Psi_S((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma_0)$ jest zdefiniowane
- $K_\Psi((A_1, \omega_1), \dots, (A_n, \omega_n)) = \sum_{i=1}^n \kappa_s(A_i, \sigma_{i-1}, \sigma_i)$, gdzie $\sigma_i = \Psi_S(A_i, \sigma_{i-1})$

wtedy dla każdego programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$, $\Pi_S((A_1, \omega_1), \dots, (A_n, \omega_n))$ jest zbiorem wszystkich mapowań Ψ_S , dla których $K_\Psi((A_1, \omega_1), \dots, (A_n, \omega_n))$ jest minimalne.

3 Język kwerend

Dla dziedziny akcji D możemy definiować zdania, zwane kwerendami.

3.1 Syntaktyka

Uwaga: We wszystkich kwerendach, podanie agenta wykonującego daną akcję nie jest wymagane. Jeżeli agent nie zostanie podany, wtedy akcja jest wykonywana przez kogokolwiek.

necessary executable $\omega_1 A_1, \dots, \omega_n A_n$

Program działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$ jest wykonywalny zawsze.

possibly executable $\omega_1 A_1, \dots, \omega_n A_n$

Program działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$ jest wykonywalny kiedykolwiek.

necessary γ after $\omega_1 A_1, \dots, \omega_n A_n$ **from** π

Wykonanie programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$ z dowolnego stanu spełniającego warunek π zawsze prowadzi do stanu spełniającego warunek γ .

possibly γ after $\omega_1 A_1, \dots, \omega_n A_n$ **from** π

Wykonanie programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$ z dowolnego stanu spełniającego warunek π kiedykolwiek prowadzi do stanu spełniającego warunek γ .

typically γ after $\omega_1 A_1, \dots, \omega_n A_n$ **from** π

Wykonanie programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$ z dowolnego stanu spełniającego warunek π na ogół prowadzi do stanu spełniającego warunek γ .

necessary accessible γ **from** π

Wykonanie pewnego programu działań z dowolnego stanu spełniającego warunek π zawsze prowadzi do stanu spełniającego warunek γ .

possibly accessible γ **from** π

Wykonanie pewnego programu działań z dowolnego stanu spełniającego warunek π kiedykolwiek prowadzi do stanu spełniającego warunek γ .

typically accessible γ **from** π

Wykonanie pewnego programu działań z dowolnego stanu spełniającego warunek π na ogół prowadzi do stanu spełniającego warunek γ .

necessary ω in $\omega_1 A_1, \dots, \omega_n A_n$

Wykonawca ω jest zawsze zaangażowany w realizację programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$.

possibly ω in $\omega_1 A_1, \dots, \omega_n A_n$

Wykonawca ω jest kiedykolwiek zaangażowany w realizację programu działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$.

3.2 Semantyka

Definicja 3.1 (Konsekwencja) Niech D będzie dziedziną akcji języka \mathcal{ADA} . Q jest konsekwencją D (ozn. $D \models Q$), wtedy i tylko wtedy kiedy:

- jeżeli Q jest postaci **necessary executable** $\omega_1 A_1, \dots, \omega_n A_n$:
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D oraz każdego mapowania Ψ_{SAb} , $\Psi_{SAb}(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma_0)$ jest zdefiniowane.
- jeżeli Q jest postaci **possibly executable** $\omega_1 A_1, \dots, \omega_n A_n$:
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , istnieje mapowanie Ψ_S , takie że $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma_0)$ jest zdefiniowane.
- jeżeli Q jest postaci **necessary γ after $\omega_1 A_1, \dots, \omega_n A_n$ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π oraz każdego mapowania Ψ_{SAb} zachodzi $\Psi_{SAb}(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **possibly γ after $\omega_1 A_1, \dots, \omega_n A_n$ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π istnieje mapowanie Ψ_S , takie że zachodzi $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **typically γ after $\omega_1 A_1, \dots, \omega_n A_n$ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π oraz każdego mapowania Ψ_{SN} zachodzi $\Psi_{SN}(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **necessary accessible γ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π oraz każdego mapowania Ψ_{SAb} istnieje program działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$, taki że $\Psi_{SAb}(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **possibly accessible γ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π , istnieje mapowanie Ψ_S , dla którego istnieje program działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$, taki że $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **typically accessible γ from π** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , każdego stanu $\sigma \in \Sigma$ spełniającego warunek π oraz każdego mapowania Ψ_{SN} istnieje program działań $((A_1, \omega_1), \dots, (A_n, \omega_n))$, taki że $\Psi_{SN}(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma) \models \gamma$.
- jeżeli Q jest postaci **necessary ω in $\omega_1 A_1, \dots, \omega_n A_n$** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D oraz każdego mapowania Ψ_S , $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma_0)$ jest zdefiniowane oraz ω jest wykonawcą w jednym z elementów ciągu $((A_1, \omega_1), \dots, (A_n, \omega_n))$ lub istnieje element tego ciągu (A_i, ε) , $1 \leq i \leq n$, taki że akcja A_i jest wykonywalna **jedynie** przez wykonawcę ω w stanie otrzymanym po wykonaniu planu działania $((A_1, \omega_1), \dots, (A_{i-1}, \omega_{i-1}))$ lub w stanie σ_0 jeżeli $i = 1$, a także $\Psi_S(((A_i, \omega), \dots, (A_n, \omega_n)), \sigma)$ jest zdefiniowane (dla każdego stanu σ osiągniętego po wykonaniu poprzednich akcji).
- jeżeli Q jest postaci **possibly ω in $\omega_1 A_1, \dots, \omega_n A_n$** :
dla każdej struktury $S = (\Sigma, \sigma_0, ResAb, ResN)$, będącej modelem D , istnieje mapowanie Ψ_S , takie że $\Psi_S(((A_1, \omega_1), \dots, (A_n, \omega_n)), \sigma_0)$ jest zdefiniowane oraz ω jest wykonawcą w jednym z elementów ciągu $((A_1, \omega_1), \dots, (A_n, \omega_n))$ lub istnieje element tego ciągu (A_i, ε) , $1 \leq i \leq n$, taki że akcja A_i jest wykonywalna przez wykonawcę ω w stanie otrzymanym po wykonaniu planu działania $((A_1, \omega_1), \dots, (A_{i-1}, \omega_{i-1}))$ lub w stanie σ_0 jeżeli $i = 1$, a także $\Psi_S(((A_i, \omega), \dots, (A_n, \omega_n)), \sigma)$ jest zdefiniowane (istnieje spełniający ten warunek stan σ osiągnięty po wykonaniu poprzednich akcji).

4 Przykład

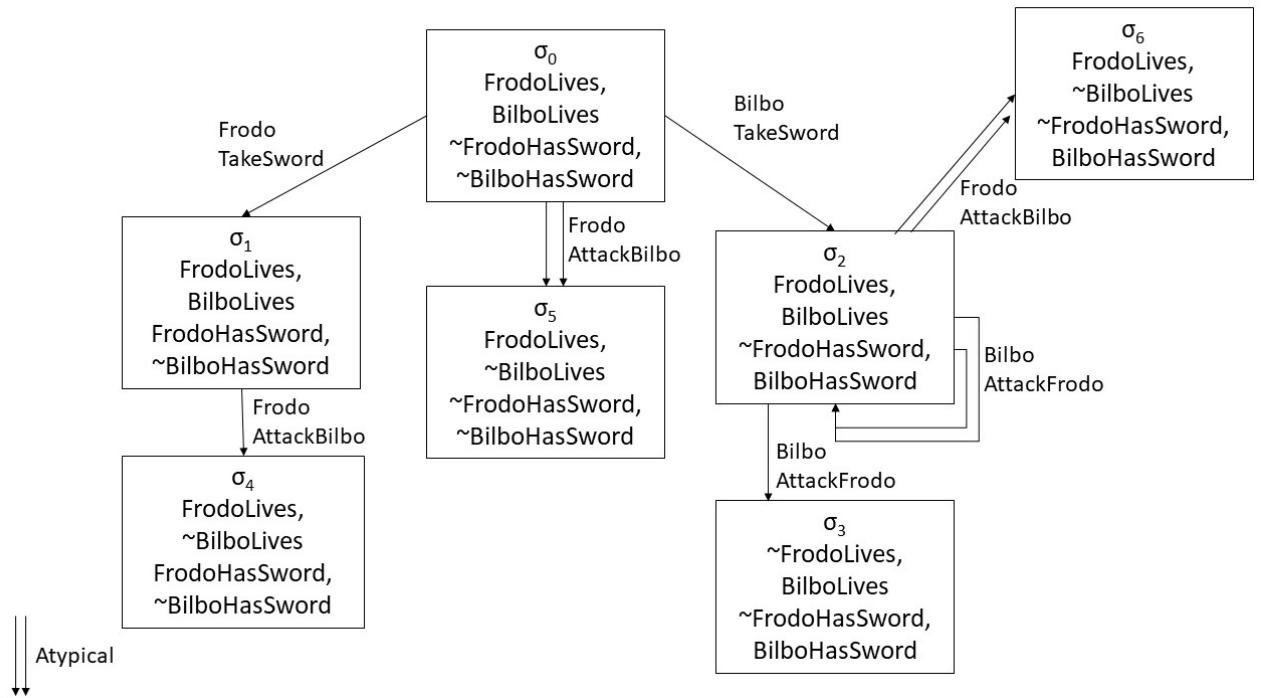
Na arenie zamknięto dwóch niziołków: Bilba i Froda. Arenę jako żywy może opuścić tylko jeden. Na arenie znajduje się miecz. Frodo jest silny i młody a Bilbo stary i słaby. Frodo jest w stanie pokonać Bilba bez użycia miecza, a Bilbo Froda tylko przy użyciu broni. Bilbo słabo włada mieczem i może się okazać, że cios wyprowadzony przez niego bronią nie zabije przeciwnika. Zakładamy, że wrogowie, nie wyrywają sobie miecza nawzajem po podniesieniu go przez któregoś z nich.

System dynamiczny opisujący powyższą sytuację posiada:

- zbiór agentów: $\Omega = \{\text{Frodo}, \text{Bilbo}\}$
- zbiór akcji: $A_c = \{\text{Take Sword}, \text{AttackFrodo}, \text{AttackBilbo}\}$
- zbiór fluentów: $\mathcal{F} = \{\text{FrodoLives}, \text{BilboLives}, \text{FrodoHasSword}, \text{BilboHasSword}\}$
- zbiór stanów: $\Sigma = \{$
 $\sigma_0 = \{\text{FrodoLives}, \text{BilboLives}, \neg\text{FrodoHasSword}, \neg\text{BilboHasSword}\},$
 $\sigma_1 = \{\text{FrodoLives}, \text{BilboLives}, \text{FrodoHasSword}, \neg\text{BilboHasSword}\},$
 $\sigma_2 = \{\text{FrodoLives}, \text{BilboLives}, \neg\text{FrodoHasSword}, \text{BilboHasSword}\},$
 $\sigma_3 = \{\neg\text{FrodoLives}, \text{BilboLives}, \neg\text{FrodoHasSword}, \text{BilboHasSword}\},$
 $\sigma_4 = \{\text{FrodoLives}, \neg\text{BilboLives}, \text{FrodoHasSword}, \neg\text{BilboHasSword}\},$
 $\sigma_5 = \{\text{FrodoLives}, \neg\text{BilboLives}, \neg\text{FrodoHasSword}, \neg\text{BilboHasSword}\},$
 $\sigma_6 = \{\text{FrodoLives}, \neg\text{BilboLives}, \neg\text{FrodoHasSword}, \text{BilboHasSword}\}$
}.

Mamy następującą domenę akcji:

```
initially FrodoLives and BilboLives and not FrodoHasSword and not BilboHasSword
when Frodo TakeSword causes FrodoHasSword if not BilboHasSword
when Bilbo TakeSword causes BilboHasSword if not FrodoHasSword
impossible Frodo AttackFrodo
impossible Frodo TakeSword if not FrodoLives
impossible Bilbo AttackBilbo
impossible Bilbo TakeSword if not BilboLives
AttackFrodo typically causes not FrodoLives if BilboHasSword
AttackBilbo causes not BilboLives if FrodoHasSword
observable not BilboLives after AttackBilbo if not FrodoHasSword
```



4.1 Kwerendy

- Czy Frodo zawsze atakuje Bilbo?

necessary executable Frodo AttackBilbo

Odpowiedź: False

- Czy możliwe jest, że Frodo zaatakuje Bilbo, po tym jak Bilbo weźmie miecz i zaatakuje Froda?

possibly executable Bilbo TakeSword, Bilbo AttackFrodo, Frodo AttackBilbo

Odpowiedź: True

- Czy Bilbo zawsze umiera, jeśli Frodo weźmie miecz i zaatakuje Bilbo, zakładając że obaj żyją?

necessary \neg BilboLives **after** Frodo TakeSword, Frodo AttackBilbo **from** BilboLives and FrodoLives and \neg FrodoHasSword

Odpowiedź: False

- Czy Frodo na ogół umiera, jeśli Bilbo weźmie miecz i zaatakuje Frodo, zakładając że Bilbo i Frodo żyją, i Frodo nie ma miecza?

typically \neg FrodoLives **after** Bilbo TakeSword, Bilbo AttackFrodo **from** BilboLives and FrodoLives and \neg FrodoHasSword

Odpowiedź: True

- Czy Bilbo kiedykolwiek umiera, jeśli weźmie miecz, zaatakuje Frodo, po czym Frodo zaatakuje jego, jeśli na początku obaj żyją?

possibly \neg BilboLives **after** Bilbo TakeSword, Bilbo AttackFrodo, Frodo AttackBilbo **from** BilboLives and FrodoLives

Odpowiedź: True

- Czy zawsze możliwe jest osiągnięcie stanu, w którym Frodo żyje po wykonaniu pewnego programu w przypadku, gdy stan początkowy spełnia warunek, że Bilbo żyje, a Frodo ma miecz?

necessary accessible FrodoLives **from** BilboLives, FrodoHasSword

Odpowiedź: True

- Czy kiedykolwiek możliwe jest osiągnięcie stanu, w którym obaj żyją po wykonaniu pewnego programu w przypadku, gdy stan początkowy spełnia warunek, że obaj żyją, a Bilbo ma miecz?

possibly accessible FrodoLives and BilboLives **from** FrodoLives and BilboLives and BilboHasSword

Odpowiedź: True

Tablica 1: Różnice między stanami

	σ_0	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
σ_0	\emptyset	{Frodo Has Sword}	{Bilbo Has Sword}	{¬Frodo Lives, Bilbo Has Sword}	{¬Bilbo Lives, Frodo Has Sword}	{¬Bilbo Lives}	{¬Bilbo Lives, Bilbo Has Sword}
σ_1	{¬Frodo Has Sword}	\emptyset	{¬Frodo Has Sword, Bilbo Has Sword}	{¬Frodo Lives, ¬Frodo Has Sword, Bilbo Has Sword}	{¬Bilbo Lives}	{¬Bilbo Lives, ¬Frodo Has Sword}	{¬Bilbo Lives, ¬Frodo Has Sword, Bilbo Has Sword}
σ_2	{¬Bilbo Has Sword}	{Frodo Has Sword, ¬Bilbo Has Sword}	\emptyset	{¬Frodo Lives}	{¬Bilbo Lives, Frodo Has Sword, ¬Bilbo Has Sword}	{¬Bilbo Lives, ¬Bilbo Has Sword}	{¬Bilbo Lives}
σ_3	{Frodo Lives, ¬Bilbo Has Sword}	{Frodo Lives, Frodo Has Sword, ¬Bilbo Has Sword}	{Frodo Lives}	\emptyset	{Frodo Lives, ¬Bilbo Lives, Frodo Has Sword, ¬Bilbo Has Sword}	{Frodo Lives, ¬Bilbo Lives, ¬Bilbo Has Sword}	{Frodo Lives, ¬Bilbo Lives}
σ_4	{Bilbo Lives, ¬Frodo Has Sword}	{Bilbo Lives}	{Bilbo Lives, ¬Frodo Has Sword, Bilbo Has Sword}	{¬Frodo Lives, Bilbo Lives, ¬Frodo Has Sword, Bilbo Has Sword}	\emptyset	{¬Frodo Has Sword}	{¬Frodo Has Sword, Bilbo Has Sword}
σ_5	{Bilbo Lives}	{Bilbo Lives, Frodo Has Sword}	{Bilbo Lives, Bilbo Has Sword}	{¬Frodo Lives, Bilbo Lives, Bilbo Has Sword}	{Frodo Has Sword}	\emptyset	{Bilbo Has Sword}
σ_6	{Bilbo Lives, ¬Bilbo Has Sword}	{Bilbo Lives, Frodo Has Sword, ¬Bilbo Has Sword}	{Bilbo Lives}	{¬Frodo Lives, Bilbo Lives}	{Frodo Has Sword, ¬Bilbo Has Sword}	{¬Bilbo Has Sword}	\emptyset

Tablica 3: Res_0 , Res_0^+ , Res^-

		Frodo TakeSword	Bilbo TakeSword	AttackBilbo	AttackFrodo
Res_0	σ_0	$\{\sigma_1, \sigma_4\}$	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_1	$\{\sigma_1, \sigma_4\}$	\emptyset	$\{\sigma_4, \sigma_5, \sigma_6\}$	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_2	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_3	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_4	$\{\sigma_1, \sigma_4\}$	\emptyset	$\{\sigma_4, \sigma_5, \sigma_6\}$	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_5	$\{\sigma_1, \sigma_4\}$	\emptyset	\emptyset	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
	σ_6	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$
Res^-	σ_0	$\{\sigma_1\}$	$\{\sigma_2\}$	\emptyset	$\{\sigma_0\}$
	σ_1	$\{\sigma_1\}$	\emptyset	$\{\sigma_4\}$	$\{\sigma_1\}$
	σ_2	\emptyset	$\{\sigma_2\}$	\emptyset	$\{\sigma_2\}$
	σ_3	\emptyset	$\{\sigma_3\}$	\emptyset	$\{\sigma_3\}$
	σ_4	$\{\sigma_4\}$	\emptyset	$\{\sigma_4\}$	$\{\sigma_4\}$
	σ_5	$\{\sigma_4\}$	\emptyset	\emptyset	$\{\sigma_5\}$
	σ_6	\emptyset	$\{\sigma_6\}$	\emptyset	$\{\sigma_6\}$
Res_0^+	σ_0	$\{\sigma_1, \sigma_4\}$	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	\emptyset
	σ_1	$\{\sigma_1, \sigma_4\}$	\emptyset	$\{\sigma_4, \sigma_5, \sigma_6\}$	\emptyset
	σ_2	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_3\}$
	σ_3	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_3\}$
	σ_4	$\{\sigma_1, \sigma_4\}$	\emptyset	$\{\sigma_4, \sigma_5, \sigma_6\}$	\emptyset
	σ_5	$\{\sigma_1, \sigma_4\}$	\emptyset	\emptyset	\emptyset
	σ_6	\emptyset	$\{\sigma_2, \sigma_3, \sigma_6\}$	\emptyset	$\{\sigma_3\}$

Tablica 4: $ResAb$, $ResN$

		Frodo TakeSword	Bilbo TakeSword	AttackBilbo	AttackFrodo
$ResN$	σ_0	$\{\sigma_1\}$	$\{\sigma_2\}$	\emptyset	\emptyset
	σ_1	$\{\sigma_1\}$	\emptyset	$\{\sigma_4\}$	\emptyset
	σ_2	\emptyset	$\{\sigma_2\}$	\emptyset	$\{\sigma_3\}$
	σ_3	\emptyset	$\{\sigma_3\}$	\emptyset	$\{\sigma_3\}$
	σ_4	$\{\sigma_4\}$	\emptyset	$\{\sigma_4\}$	\emptyset
	σ_5	$\{\sigma_4\}$	\emptyset	\emptyset	\emptyset
	σ_6	\emptyset	$\{\sigma_6\}$	\emptyset	$\{\sigma_3\}$
$ResAb$	σ_0	\emptyset	\emptyset	\emptyset	$\{\sigma_0\}$
	σ_1	\emptyset	\emptyset	\emptyset	$\{\sigma_1\}$
	σ_2	\emptyset	\emptyset	\emptyset	$\{\sigma_2\}$
	σ_3	\emptyset	\emptyset	\emptyset	\emptyset
	σ_4	\emptyset	\emptyset	\emptyset	$\{\sigma_4\}$
	σ_5	\emptyset	\emptyset	\emptyset	$\{\sigma_5\}$
	σ_6	\emptyset	\emptyset	\emptyset	$\{\sigma_6\}$