

REPREZENTACJA WIEDZY  
PROJEKT NR 4

---

Realizacja efektywnych  
programów działań

---

*Zespół:*

Adam KOMOROWSKI

Krzysztof ADAMIAK

Krzysztof KACHNIARZ

Maciej KACHNIARZ

Mateusz KIŁOS

Tomasz LUŚTYK

Paweł PARADOWSKI

Katarzyna ŻUK

16 kwietnia 2014

# Spis treści

<b>1</b>	<b>Przedstawienie problemu</b>	<b>2</b>
<b>2</b>	<b>Syntaktyka języka</b>	<b>3</b>
2.1	Język akcji . . . . .	3
<b>3</b>	<b>Semantyka języka</b>	<b>6</b>
<b>4</b>	<b>Język kwerend</b>	<b>9</b>
4.1	Q1 . . . . .	9
4.1.1	Czy program działań jest zawsze realizowalny? . . . . .	9
4.1.2	Czy program działań jest kiedykolwiek realizowalny? . . . . .	10
4.2	Q2 . . . . .	11
4.2.1	Czy program działań zawsze prowadzi do osiągnięcia celu $\gamma$ ? . . . . .	11
4.2.2	Czy program działań kiedykolwiek prowadzi do osiągnięcia celu $\gamma$ ? . . . . .	12
4.3	Q3 . . . . .	13
4.3.1	Przy jakim koszcie cel $\gamma$ jest zawsze osiągalny? . . . . .	13
4.3.2	Przy jakim koszcie cel $\gamma$ jest kiedykolwiek osiągalny? . . . . .	13
<b>5</b>	<b>Przykłady</b>	<b>15</b>
5.1	Przykład 1 . . . . .	15
5.2	Przykład 2 . . . . .	20
5.3	Przykład 3 . . . . .	22
5.4	Przykład 4 . . . . .	26
5.5	Przykład 5 . . . . .	28
<b>6</b>	<b>Wygląd programu</b>	<b>31</b>
6.1	Zakładka <i>Historia</i> . . . . .	31
6.2	Zakładka <i>Zapytania</i> . . . . .	32
6.3	Zakładka <i>Rezultat</i> . . . . .	35

# 1 Przedstawienie problemu

Celem projektu było zaprojektowanie klasy systemów dynamicznych spełniających następujące warunki:

1. Prawo inercji.
2. Pełna informacja o wszystkich akcjach i wszystkich skutkach bezpośrednich.
3. Sekwencyjność i niedeterminizm działań.
4. Z każdą akcją związany jest
  - warunek wstępny,
  - warunek wynikowy,
  - koszt realizacji.
5. Skutki akcji zależą od
  - stanu początkowego jej wykonania,
  - kosztu jej realizacji.
6. W pewnych stanach akcja może być niewykonalna.
7. Warunki integralności wpływają na wykonalność akcji.
8. Dopuszczalny jest opis częściowy zarówno stanu początkowego, jak i pewnych stanów wynikających z wykonania sekwencji akcji.

## 2 Syntaktyka języka

Mamy dane następujące zbiory:

- $Ac$ , oznaczający zbiór wszystkich akcji.
- $F$ , oznaczający zbiór wszystkich fluentów.
- $F_I \subseteq F$ , oznaczający zbiór wszystkich fluentów inertnych.
- $P$ , zbiór możliwych stanów zasobu.
- Dla każdego  $A_i \in Ac$  istnieje skończony zbiór  $C_{A_i} \subseteq P$ , oznaczający zbiór cen akcji  $A_i$ . Zbiór  $\{C_{A_i} : A_i \in Ac\}$  będziemy oznaczać jako  $C$ .

Nasz język będzie charakteryzowany za pomocą sygnatury:  $(Ac, F, P, C)$ .

**Definicja 2.1.** *Formułą* nazywamy wyrażenie spełniające warunki:

- $\top$  i  $\perp$  są formułami (pierwsza oznacza formułę zawsze prawdziwą, druga formułę zawsze fałszywą).
- Jeśli  $f \in F$ , to  $f$  jest formułą;
- Jeśli  $\alpha$  jest formułą, to  $\neg\alpha$  jest formułą;
- Jeśli  $\alpha, \beta$  są formułami, to  $\alpha \vee \beta$  jest formułą;
- Jeśli  $\alpha, \beta$  są formułami, to  $\alpha \wedge \beta$  jest formułą;
- Jeśli  $\alpha, \beta$  są formułami, to  $\alpha \rightarrow \beta$  jest formułą;
- Jeśli  $\alpha, \beta$  są formułami, to  $\alpha \leftrightarrow \beta$  jest formułą;

### 2.1 Język akcji

Wyrażenia naszego języka akcji są postaci:

1.  $\alpha$  **after**  $A_1, \dots, A_n$ , wyrażenie to oznacza, że wykonanie ciągu akcji:  $A_1, \dots, A_n$  ze stanu początkowego z początkowym stanem zasobu, prowadzi zawsze do osiągnięcia stanu spełniającego warunek logiczny  $\alpha$ .
2. **initially**  $\alpha$ , równoważne:  $\alpha$  **after**  $A_1, \dots, A_n$  z  $n = 0$ , wyrażenie to oznacza że stan początkowy spełnia warunek logiczny  $\alpha$ .

3. **obsevable  $\alpha$  after  $A_1, \dots, A_n$** , wyrażenie to oznacza, że wykonanie ciągu akcji:  $A_1, \dots, A_n$  ze stanu początkowego z początkowym stanem zasobu, może prowadzić (prowadzi czasem) do osiągnięcia stanu spełniającego warunek logiczny  $\alpha$ .
4.  **$A$  costs  $p$  if  $\pi$  with effect  $\alpha$** , wyrażenie to oznacza, że wykonanie akcji  $A$  ze stanu spełniającego warunek logiczny  $\pi$  przy koszcie akcji  $p$  prowadzi do osiągnięcia stanu spełniającego warunek logiczny  $\alpha$ .
5.  **$A$  costs  $p$  with effect  $\alpha$** , wyrażenie to jest równoważne wyrażeniu  **$A$  costs  $p$  if  $\top$  with effect  $\alpha$** , co oznacza że wykonanie akcji  $A$  z dowolnego stanu przy koszcie akcji  $p$  prowadzi do osiągnięcia stanu spełniającego warunek logiczny  $\alpha$ .
6.  **$A$  costs  $p$  if  $\pi$** , wyrażenie oznacza, że wykonanie akcji  $A$  ze stanu spełniającego warunek logiczny  $\pi$  jest wykonywane kosztem akcji co najmniej  $p$ .
7.  **$A$  costs  $p$** , wyrażenie to jest równoważne:  **$A$  costs  $p$  if  $\top$** , co oznacza, że wykonanie akcji  $A$  z dowolnego stanu jest wykonywane kosztem akcji co najmniej  $p$ .
8.  **$A$  if  $\pi$  with effect  $\alpha$** , wyrażenie to oznacza, że wykonanie akcji  $A$  na stanie spełniającym warunek logiczny  $\pi$ , prowadzi do stanu spełniającego warunek:  $\alpha$  przy zerowym koszcie.
9.  **$A$  with effect  $\alpha$** , wyrażenie to jest równoważne:  **$A$  causes  $\alpha$  if  $\top$** , co oznacza, że wykonanie akcji  $A$  w dowolnym stanie prowadzi do stanu spełniającego warunek:  $\alpha$  przy zerowym koszcie.
10. **impossible  $A$  if  $\pi$** , wyrażenie to jest równoważne:  **$A$  if  $\pi$  with effect  $\perp$** , co oznacza, że wykonanie akcji  $A$  w stanie spełniającym  $\pi$  jest niemożliwe.
11.  **$A$  releases  $f$  if  $\pi$** , wyrażenie to oznacza, że wykonanie akcji  $A$  na stanie spełniającym warunek logiczny  $\pi$  może (ale nie musi) doprowadzić do zmiany wartości fluentu inertnego  $f$ .
12.  **$A$  releases  $f$** , wyrażenie to jest równoważne  **$A$  releases  $f$  if  $\top$** , co oznacza, że wykonanie akcji  $A$  na dowolnym stanie może (ale nie musi) doprowadzić do zmiany wartości fluentu inertnego  $f$ .
13. **always  $\alpha$** , wyrażenie to oznacza, że dowolny stan układu spełnia warunek logiczny  $\alpha$ .

14. **noninertial**  $f$ , wyrażenie to oznacza, że fluent  $f$  jest nieinertny.
15.  $A$  **preserves**  $f$  **if**  $\pi$ , wyrażenie to oznacza, że wykonanie akcji  $A$  w stanie spełniającym warunek logiczny  $\pi$  nie może zmienić wartości fluentu  $f$ .
16. **initially**  $p$  **value**, wyrażenie to oznacza, że początkowy stan zasobu wynosi  $p$ .

### 3 Semantyka języka

**Definicja 3.1.** *Stanem* nazywamy funkcję:  $\sigma : F \rightarrow \{0, 1\}$ .

**Uwaga 3.1.** Definicję stanu można rozszerzyć na zbiór wszystkich formuł nad zbiorem fluentów  $F$  do funkcji  $\sigma^*$  w następujący sposób:

- $\sigma^*(\top) = 1$ .
- $\sigma^*(\perp) = 0$ .
- $\sigma^*(f) = \sigma(f)$ .
- $\sigma^*(\neg\alpha) = 1 - \sigma^*(\alpha)$ .
- $\sigma^*(\alpha \vee \beta) = \max(\sigma^*(\alpha), \sigma^*(\beta))$ .
- $\sigma^*(\alpha \wedge \beta) = \min(\sigma^*(\alpha), \sigma^*(\beta))$ .
- $\sigma^*(\alpha \rightarrow \beta) = \sigma^*(\neg\alpha \vee \beta)$ .
- $\sigma^*(\alpha \leftrightarrow \beta) = \sigma^*((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha))$ .

gdzie  $\alpha, \beta$  są formułami nad  $F$  i  $f \in F$ .

Jeśli  $\sigma^*(\alpha) = 1$ , to piszemy:  $\sigma \models \alpha$ .

**Definicja 3.2.** *Strukturą* naszego języka jest piątka:  $S = (\Sigma, P, \sigma_0, p_0, Res)$  taka, że:

1.  $\Sigma \neq \emptyset$  jest podzbiorem wszystkich stanów.
2.  $\sigma_0$  jest stanem początkowym układu.
3.  $p_0$  jest początkowym stanem zasobu.
4.  $Res : \Sigma \times P \times Ac \rightarrow 2^{\Sigma \times P}$  zwana funkcją tranzycji.

**Definicja 3.3.** Na strukturze  $S = (\Sigma, P, \sigma_0, p_0, Res)$  definiujemy częściową funkcję  $\Psi_S : Ac^* \times \Sigma \times P \rightarrow \Sigma \times P$  spełniającą następujące warunki:

- $\Psi_S(\epsilon, \sigma, p) = (\sigma, p)$ .
- Jeśli dla  $n \geq 1$ ,  $\Psi_S((A_1, \dots, A_n), \sigma, p)$  jest zdefiniowane,
- Jeżeli  $Res((A_1, \dots, A_n, A_{n+1}), \sigma, p) \neq \emptyset$ , to  
 $\Psi_S((A_1, \dots, A_n, A_{n+1}), \sigma, p) \in Res(A_{n+1}, \Psi_S((A_1, \dots, A_n), \sigma, p))$ .

**Definicja 3.4.** Podamy pewne podstawowe definicje:

- Fluent  $f$  jest inertny w  $D$  wtedy i tylko wtedy gdy  $(\mathbf{noninertial} f) \notin D$
- $\sigma : F \rightarrow \{0, 1\}$  jest stanem w  $D$ , jeśli dla każdego wyrażenia  $(\mathbf{always} \alpha) \in D$  zachodzi  $\sigma \models \alpha$ .
- Wyrażenie  $\alpha$  **after**  $A_1, \dots, A_n$  jest prawdziwe w  $S$  wtedy i tylko wtedy gdy, dla dowolnej funkcji  $\Psi_S$  zdefiniowanej wcześniej  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0) \models \alpha$ .
- Wyrażenie **observable**  $\alpha$  **after**  $A_1, \dots, A_n$  jest prawdziwe w  $S$  wtedy i tylko wtedy gdy, dla pewnej funkcji  $\Psi_S$  zdefiniowanej wcześniej  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0) \models \alpha$ .
- Wyrażenie **initially**  $p$  **value** jest prawdziwe w  $S$  wtedy i tylko wtedy gdy,  $p_0 = p$ .

**Definicja 3.5.** Niech  $D$  będzie domeną akcyjną i  $S = (\Sigma, P, \sigma_0, p_0, Res)$  będzie strukturą naszego języka. Wtedy możemy zdefiniować przekształcenie:  $Res_0 : Ac \times \Sigma \times P \rightarrow 2^{\Sigma \times P}$ , takie że dla dowolnej funkcji  $A \in Ac$ , dla dowolnego  $p \in P$  i dla dowolnego stanu  $\sigma \in \Sigma$ :

$$Res_0(A, \sigma, p) = \{(\sigma', p) \in \Sigma \times P : ((A \text{ causes } \alpha \text{ if } \pi) \in D \wedge \sigma \models \pi) \Rightarrow \sigma' \models \alpha\} \cup \{(\sigma', p - p_A) \in \Sigma \times P : ((A \text{ cost } p \text{ if } \pi \text{ with effect } \alpha) \in D \wedge \sigma \models \pi) \Rightarrow \sigma' \models \alpha) \wedge p_A \in C_A\}.$$

**Uwaga 3.2.** Dodatkowo dla każdej pary stanów  $\sigma, \sigma' \in \Sigma$  i dla dowolnej akcji  $A \in Ac$  definiujemy zbiór

$$New(A, \sigma, \sigma') = \{\bar{f} : \sigma' \models \bar{f} \wedge ((f \in F_I \wedge \sigma(f) \neq \sigma'(f)) \vee (\text{Dla pewnego wyrażenia } (A \text{ releases } f \text{ if } \pi) \in D \text{ zachodzi } \sigma \models \pi))\}.$$

**Definicja 3.6.** Oznaczmy  $D^-$  jako  $D$  bez wszystkich wyrażeń postaci:  $A$  **preserves**  $f$  **if**  $\pi$ . Niech  $S = (\Sigma, P, \sigma_0, p_0, Res_{RW})$  będzie strukturą naszego języka. Mówimy, że  $S$  jest modelem w  $D^-$  wtedy i tylko wtedy kiedy zachodzą warunki:

- $\Sigma$  jest zbiorem wszystkich stanów w  $D^-$ .
- Każde wyrażenie postaci:  $(\alpha \text{ after } A_1, \dots, A_n)$ ,  $(\mathbf{observable} \alpha \text{ after } A_1, \dots, A_n)$  i  $(\mathbf{initially} p \text{ value})$  z  $D^-$  są prawdziwe w  $S$ .
- Dla każdego  $A \in Ac$ , dla każdego  $\sigma \in \Sigma$  i dla każdego  $p \in P$ ,

$$Res_{RW}(A, \sigma, p) = \{(\sigma', p') \in Res_0 : New(A, \sigma, \sigma') \text{ jest minimalny}\}.$$



Zbiór takich modeli oznaczamy symbolem:  $Mod(D^-)$ .

**Definicja 3.7.** Struktura  $S = (\Sigma, P, \sigma_0, p_0, Res) \in Mod(D^-)$  jest modelem w D wtedy i tylko wtedy gdy dla dowolnej akcji  $A \in Ac$ , dla dowolnego stanu  $\sigma \in \Sigma$  i dla dowolnego stanu zasobu  $p \in P$ :

$$Res(A, \sigma, p) = \{(\sigma', p') \in Res_{RW} : (A \text{ preserves } f \text{ if } \pi) \in D \wedge (\sigma \models \pi \Rightarrow (\sigma(f) = \sigma'(f)))\}.$$

## 4 Język kwerend

### 4.1 Q1

Kwerenda odpowiada na pytanie, czy podany program działań  $A_1, \dots, A_n$  jest realizowalny zawsze/kiedykolwiek z każdego stanu spełniającego warunek  $\pi$  przy podanym/dowolnym koszcie.

#### 4.1.1 Czy program działań jest zawsze realizowalny?

- Jeżeli kwerenda  $Q$  jest postaci:

**always executable  $A_1, \dots, A_n$  from  $\pi$  with  $p$  for  $c$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$  i dla każdego odwzorowania  $\Psi_S$ , jeżeli  $c \leq p$  to suma kosztów wykonania akcji  $A_1, \dots, A_n$  oraz  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p)$  jest definiowalne.

Innymi słowy – zawsze w każdym możliwym stanie spełniającym  $\pi$  i z zasobem wielkości  $p$  możemy wykonać ciąg akcji  $A_1, \dots, A_n$ , którego koszt jest równy  $c$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**always executable  $A_1, \dots, A_n$  from  $\pi$  with  $p$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$  i dla każdego odwzorowania  $\Psi_S$ , jeżeli  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p)$  jest definiowalne.

Innymi słowy – zawsze w każdym możliwym stanie spełniającym  $\pi$  i z zasobem wielkości  $p$  możemy wykonać ciąg akcji  $A_1, \dots, A_n$  o dowolnym koszcie.

- Jeżeli kwerenda  $Q$  jest postaci:

**always executable  $A_1, \dots, A_n$  for  $c$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego odwzorowania  $\Psi_S$  i dla  $c \leq p_0$  równego sumie kosztów wykonania akcji  $A_1, \dots, A_n$  –  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0)$  jest definiowalne.

Innymi słowy – zawsze ze stanu początkowego możemy wykonać ciąg akcji  $A_1, \dots, A_n$ , którego koszt jest równy  $c$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**always executable**  $A_1, \dots, A_n$ ,

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego odwzorowania  $\Psi_S - \Psi_S((A_1, \dots, A_n), \sigma_0, p_0)$  jest definiowalne.

Innymi słowy – zawsze ze stanu początkowego możemy wykonać ciąg akcji  $A_1, \dots, A_n$  o dowolnym koszcie.

#### 4.1.2 Czy program działań jest kiedykolwiek realizowalny?

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes executable**  $A_1, \dots, A_n$  **from**  $\pi$  **with**  $p$  **for**  $c$ ,

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$  i dla pewnego odwzorowania  $\Psi_S$ , jeżeli  $c \leq p$  to suma kosztów wykonania akcji  $A_1, \dots, A_n$  oraz  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p)$  jest definiowalne.

Innymi słowy – czasami możliwe jest wykonanie ciągu akcji  $A_1, \dots, A_n$  o koszcie równym  $c$  z dowolnego stanu spełniającego  $\pi$  i z zasobem równym  $p$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes executable**  $A_1, \dots, A_n$  **from**  $\pi$  **with**  $p$ ,

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$  i dla pewnego odwzorowania  $\Psi_S$ , jeżeli  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p)$  jest definiowalne.

Innymi słowy – czasami możliwe jest wykonanie ciągu akcji  $A_1, \dots, A_n$  o dowolnym koszcie z dowolnego stanu spełniającego  $\pi$  i z zasobem równym  $p$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes executable  $A_1, \dots, A_n$  for  $c$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla pewnego odwzorowania  $\Psi_S$  i dla  $c \leq p_0$  równego sumie kosztów wykonania akcji  $A_1, \dots, A_n - \Psi_S((A_1, \dots, A_n), \sigma_0, p_0)$  jest definiowalne.

Innymi słowy – czasami możliwe jest wykonanie ciągu akcji  $A_1, \dots, A_n$  o koszcie równym  $c$  ze stanu początkowego.

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes executable  $A_1, \dots, A_n$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla pewnego odwzorowania  $\Psi_S - \Psi_S((A_1, \dots, A_n), \sigma_0, p_0)$  jest definiowalne.

Innymi słowy – czasami możliwe jest wykonanie ciągu akcji  $A_1, \dots, A_n$  o dowolnym koszcie ze stanu początkowego.

## 4.2 Q2

Kwerenda odpowiada na pytanie, czy podany program działań  $A_1, \dots, A_n$  prowadzi zawsze/kiedykolwiek do osiągnięcia celu  $\gamma$  ze stanu początkowego/spełniającego warunek  $\pi$ .

### 4.2.1 Czy program działań zawsze prowadzi do osiągnięcia celu $\gamma$ ?

- Jeżeli kwerenda  $Q$  jest postaci:

**necessary  $\gamma$  after  $A_1, \dots, A_n$  from  $\pi$  with  $p$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$ , dla każdego odwzorowania  $\Psi_S$ , jeżeli  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p) \models \gamma$ .

Innymi słowy – ciąg akcji  $A_1, \dots, A_n$  zawsze prowadzi z dowolnego stanu spełniającego  $\pi$  o zasobie  $p$  do stanu spełniającego  $\gamma$ . W przypadku  $p = 0$  oznacza to, że podany ciąg prowadzi do osiągnięcia  $\gamma$  z dowolnym stanem zasobu.

- Jeżeli kwerenda  $Q$  jest postaci:

**necessary  $\gamma$  after  $A_1, \dots, A_n$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$  oraz dla każdego odwzorowania  $\Psi_S$  zachodzi:

$$\Psi_S((A_1, \dots, A_n), \sigma_0, p_0) \models \gamma.$$

Oznacza to, że podany program działań zawsze prowadzi ze stanu początkowego do celu  $\gamma$ .

#### 4.2.2 Czy program działań kiedykolwiek prowadzi do osiągnięcia celu $\gamma$ ?

- Jeżeli kwerenda  $Q$  jest postaci:

**possibly  $\gamma$  after  $A_1, \dots, A_n$  from  $\pi$  with  $p$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$  i dla pewnego odwzorowania  $\Psi_S$ , jeżeli  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, p) \models \gamma$ .

Innymi słowy – ciąg akcji  $A_1, \dots, A_n$  czasami prowadzi z dowolnego stanu spełniającego  $\pi$  o zasobie  $p$  do stanu spełniającego  $\gamma$ . W przypadku  $p = 0$  oznacza to, że podany ciąg prowadzi do osiągnięcia  $\gamma$  z dowolnym stanem zasobu.

- Jeżeli kwerenda  $Q$  jest postaci:

**possibly  $\gamma$  after  $A_1, \dots, A_n$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$  i dla pewnego odwzorowania  $\Psi_S$  zachodzi:

$$\Psi_S((A_1, \dots, A_n), \sigma_0, p_0) \models \gamma.$$

Oznacza to, że podany program działań czasami prowadzi ze stanu początkowego do celu  $\gamma$ .

### 4.3 Q3

Kwerenda odpowiada na pytanie, przy jakim koszcie cel  $\gamma$  jest zawsze lub kiedykolwiek osiągalny ze stanu początkowego lub spełniającego warunek  $\pi$ .

#### 4.3.1 Przy jakim koszcie cel $\gamma$ jest zawsze osiągalny?

- Jeżeli kwerenda  $Q$  jest postaci:

**always cost of  $\gamma$  from  $\pi$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$ , dla każdego odwzorowania  $\Psi_S$  istnieje taki program działań  $(A_1, \dots, A_n)$ , że jeżeli  $\sigma \models \pi$ , to:

$\Psi_S((A_1, \dots, A_n), \sigma, \infty)$  jest definiowalna i  $\Psi_S((A_1, \dots, A_n), \sigma, \infty) \models \gamma$ .

Odpowiedzią na powyższą kwerendę jest maksymalna możliwa suma kosztów wykonania znalezionych akcji  $A_1, \dots, A_n$  z każdego ze stanów spełniających  $\pi$  i prowadzących do  $\gamma$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**always cost of  $\gamma$ ,**

wtedy  $D \models Q$  wtw. dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0 = \infty, Res)$  dziedziny  $D$  i dla każdego odwzorowania  $\Psi_S$  istnieje taki program działań  $(A_1, \dots, A_n)$ , że zachodzi:  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0 = \infty)$  jest definiowalna oraz  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0 = \infty) \models \gamma$ .

Odpowiedzią na powyższą kwerendę jest maksymalna możliwa suma kosztów wykonania ze stanu początkowego znalezionych akcji  $A_1, \dots, A_n$ , które prowadzą do  $\gamma$ .

#### 4.3.2 Przy jakim koszcie cel $\gamma$ jest kiedykolwiek osiągalny?

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes cost of  $\gamma$  from  $\pi$ ,**

wtedy  $D \models Q$  wtw. istnieje program działań  $(A_1, \dots, A_n)$  taki, że dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0, Res)$  dziedziny  $D$ , dla każdego stanu  $\sigma \in \Sigma$ , dla pewnego odwzorowania  $\Psi_S$ , jeżeli  $\sigma \models \pi$ , to  $\Psi_S((A_1, \dots, A_n), \sigma, \infty)$  jest definiowalna oraz zachodzi:  $\Psi_S((A_1, \dots, A_n), \sigma, \infty) \models \gamma$ .

Odpowiedzią na powyższą kwerendę jest suma kosztów wykonania znalezionej sekwencji akcji  $A_1, \dots, A_n$ , który czasami jest możliwy do wykonania z dowolnego stanu spełniającego  $\pi$ .

- Jeżeli kwerenda  $Q$  jest postaci:

**sometimes cost of  $\gamma$ ,**

wtedy  $D \models Q$  wtw. istnieje program działań  $(A_1, \dots, A_n)$  taki, że dla każdego modelu  $S = (\Sigma, P, \sigma_0, p_0 = \infty, Res)$  dziedziny  $D$  i dla pewnego odwzorowania  $\Psi_S$  zachodzi:  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0 = \infty)$  jest definiowalna oraz  $\Psi_S((A_1, \dots, A_n), \sigma_0, p_0 = \infty) \models \gamma$

Odpowiedzią na powyższą kwerendę jest suma kosztów wykonania znalezionej sekwencji akcji  $A_1, \dots, A_n$ , który czasami jest możliwy do wykonania ze stanu początkowego.

## 5 Przykłady

### 5.1 Przykład 1

Rozważmy następującą historię:

W średniowiecznym królestwie mieszkał sobie głodny murarz. Miał żonę i dużo dzieci. Pewnego jesiennego, deszczowego dnia murarz stracił pracę i był z tego powodu bardzo smutny i był już smutny całą zimę. Zdał sobie wtedy sprawę, że całą zimę będą musieli (on i jego rodzina) przeżyć tylko za 5 złotych monet (wtedy było to trochę więcej niż dzisiaj, ale też mało). W tych czasach kupienie jedzenia na całą zimę dla murarza lub jego żony kosztowało 2 złote monety. Wykarmienie wszystkich dzieci w ciągu zimy też kosztowało 2 złote monety. Kupienie jedzenia sprawiało, że ten dla którego kupiono jedzenie nie był głodny tej zimy Czy murarz zdołał wykarmić całą rodzinę tej zimy?

Ten problem można przedstawić w postaci następującego zagadnienia w naszym języku:

1. **initially** MurarzGłodny
2. **always** MurarzSmutny
3. **initially** 5 **value**
4. KUP\_JEDZENIE\_MURARZOWI *costs 2 with effect*  $\neg$  MurarzGłodny
5. KUP\_JEDZENIE\_ŻONIE *costs 2 with effect*  $\neg$  ŻonaGłodna
6. KUP\_JEDZENIE\_DZIECIOM *costs 2 with effect*  $\neg$  DzieciGłodne

gdzie jako:

1. zbiór fluentów przyjmujemy:  $F = \{ \text{MurarzSmutny}, \text{MurarzGłodny}, \text{ŻonaGłodna}, \text{DzieciGłodne} \}$
2. zbiór akcji przyjmujemy:  $Ac = \{ \text{KUP\_JEDZENIE\_MURARZOWI}, \text{KUP\_JEDZENIE\_ŻONIE}, \text{KUP\_JEDZENIE\_DZIECIOM} \}$
3. zbiorem możliwych stanów zasobu jest  $P = \mathbb{N}$
4. zbiory  $C_{\text{KUP\_JEDZENIE\_MURARZOWI}} = C_{\text{KUP\_JEDZENIE\_ŻONIE}} = C_{\text{KUP\_JEDZENIE\_DZIECIOM}} = \{2\}$



Łatwo zauważyć, że wszystkich stanów jakie uda się utworzyć ze zbioru fluentów jest  $2^4$ , przy czym w naszym modelu dopuszczalnych jest tylko  $2^3$  możliwości, co jest spowodowane tym, że w zapisie naszego problemu występuje: *always* MurarzSmutny.

Zastanówmy się nad wyglądem stanów początkowych. Otóż jest ich  $2^2$ . Są to stany:

1.  $\sigma_1 = (\text{MurarzSmutny}, \text{MurarzGłodny}, \text{ŻonaGłodna}, \text{DzieciGłodne})$
2.  $\sigma_2 = (\text{MurarzSmutny}, \text{MurarzGłodny}, \neg \text{ŻonaGłodna}, \text{DzieciGłodne})$
3.  $\sigma_3 = (\text{MurarzSmutny}, \text{MurarzGłodny}, \text{ŻonaGłodna}, \neg \text{DzieciGłodne})$
4.  $\sigma_4 = (\text{MurarzSmutny}, \text{MurarzGłodny}, \neg \text{ŻonaGłodna}, \neg \text{DzieciGłodne})$

wynika to z faktu, że w naszym programie występują wyrażenia: *initially* MurarzGłodny i *always* MurarzSmutny.

Resztę stanów (stany nie będące stanami początkowymi i należące do zbioru wszystkich stanów dopuszczalnych  $\Sigma$ ) oznaczmy jako:

1.  $\sigma_5 = (\text{MurarzSmutny}, \neg \text{MurarzGłodny}, \text{ŻonaGłodna}, \text{DzieciGłodne})$
2.  $\sigma_6 = (\text{MurarzSmutny}, \neg \text{MurarzGłodny}, \neg \text{ŻonaGłodna}, \text{DzieciGłodne})$
3.  $\sigma_7 = (\text{MurarzSmutny}, \neg \text{MurarzGłodny}, \text{ŻonaGłodna}, \neg \text{DzieciGłodne})$
4.  $\sigma_8 = (\text{MurarzSmutny}, \neg \text{MurarzGłodny}, \neg \text{ŻonaGłodna}, \neg \text{DzieciGłodne})$

Powstaje pytanie: dlaczego nie rozpartujemy stanów w których fluent MurarzSmutny jest zaprzeczony? Ponieważ stany te są niedopuszczalne (nie spełniają warunku *always* MurarzSmutny), który jest sprawdzany już na początku działania programu.

Zastanówmy się jakie stany możemy otrzymać wykonując w stanie oznaczonym przez nas jako  $\sigma_3$  akcję KUP\_JEDZENIE\_DZIECIOM i akcję KUP\_JEDZENIE\_MURARZOWI. Zaczniemy od: KUP\_JEDZENIE\_MURARZOWI. Robimy to w następujący sposób:

- Liczymy:
$$Res_0(\text{KUP\_JEDZENIE\_MURARZOWI}, \sigma_3, 5) = \{(\sigma_5, 3), (\sigma_6, 3), (\sigma_7, 3), (\sigma_8, 3)\}$$
- Dla wszystkich stanów z  $Res_0$  liczymy wartości zbiorów *New* otrzymujemy:
  - $New(\text{KUP\_JEDZENIE\_MURARZOWI}, \sigma_3, \sigma_5) = \{\text{MurarzGłodny}, \text{DzieciGłodne}\}$

- $New(KUP\_JEDZENIE\_MURARZOWI, \sigma_3, \sigma_6) = \{\text{MurarzGłodny}, \text{ŻonaGłodna}, \text{DzieciGłodne}\}$
- $New(KUP\_JEDZENIE\_MURARZOWI, \sigma_3, \sigma_7) = \{\text{MurarzGłodny}\}$
- $New(KUP\_JEDZENIE\_MURARZOWI, \sigma_3, \sigma_8) = \{\text{MurarzGłodny}, \text{ŻonaGłodna}\}$

- Szukamy który ze zbiorów  $New$  jest minimalny ze względu na zawieranie
- Liczymy  $Res_{RW}(KUP\_JEDZENIE\_MURARZOWI, \sigma_3, 5) = \{(\sigma_7, 3)\}$
- Liczymy  $Res(KUP\_JEDZENIE\_MURARZOWI, \sigma_3, 5) = \{(\sigma_7, 3)\}$

Stąd wnioskujemy, że po wykonaniu akcji  $KUP\_JEDZENIE\_MURARZOWI$  przy stanie zasobu 5 i ze stanu  $\sigma_3$  osiągamy stan  $\sigma_7$

W przypadku akcji  $KUP\_JEDZENIE\_DZIECIOM$  otrzymujemy:

- Liczymy:

$$Res_0(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, 5) = \{(\sigma_3, 3), (\sigma_4, 3), (\sigma_7, 3), (\sigma_8, 3)\}$$

- Dla wszystkich stanów z  $Res_0$  liczymy wartości zbiorów  $New$  otrzymujemy:
  - $New(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, \sigma_3) = \emptyset$
  - $New(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, \sigma_4) = \{\text{ŻonaGłodna}\}$
  - $New(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, \sigma_7) = \{\text{MurarzGłodny}\}$
  - $New(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, \sigma_8) = \{\text{MurarzGłodny}, \text{ŻonaGłodna}\}$
- Szukamy który ze zbiorów  $New$  jest minimalny ze względu na zawieranie
- Liczymy  $Res_{RW}(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, 5) = \{(\sigma_3, 3)\}$
- Liczymy  $Res(KUP\_JEDZENIE\_DZIECIOM, \sigma_3, 5) = \{(\sigma_3, 3)\}$

Stąd jeśli wykonamy akcję  $KUP\_JEDZENIE\_DZIECIOM$  na stanie  $\sigma_3$  przy wartości portfela 5 to nie opuścimy tego stanu.

Podane wyżej rozważania można przeprowadzić dla dowolnego stanu i dowolnej akcji, otrzymując jakie stany zostaną przyjęte po zastosowaniu akcji w jakimś stanie przy ustalonym stanie zasobu. Wyniki takiego postępowania

można by przedstawić na przykład na grafie o  $2^3$  wierzchołkach i krawędziach indeksowanych nazwami akcjami (i ewentualnie ceną tych akcji).

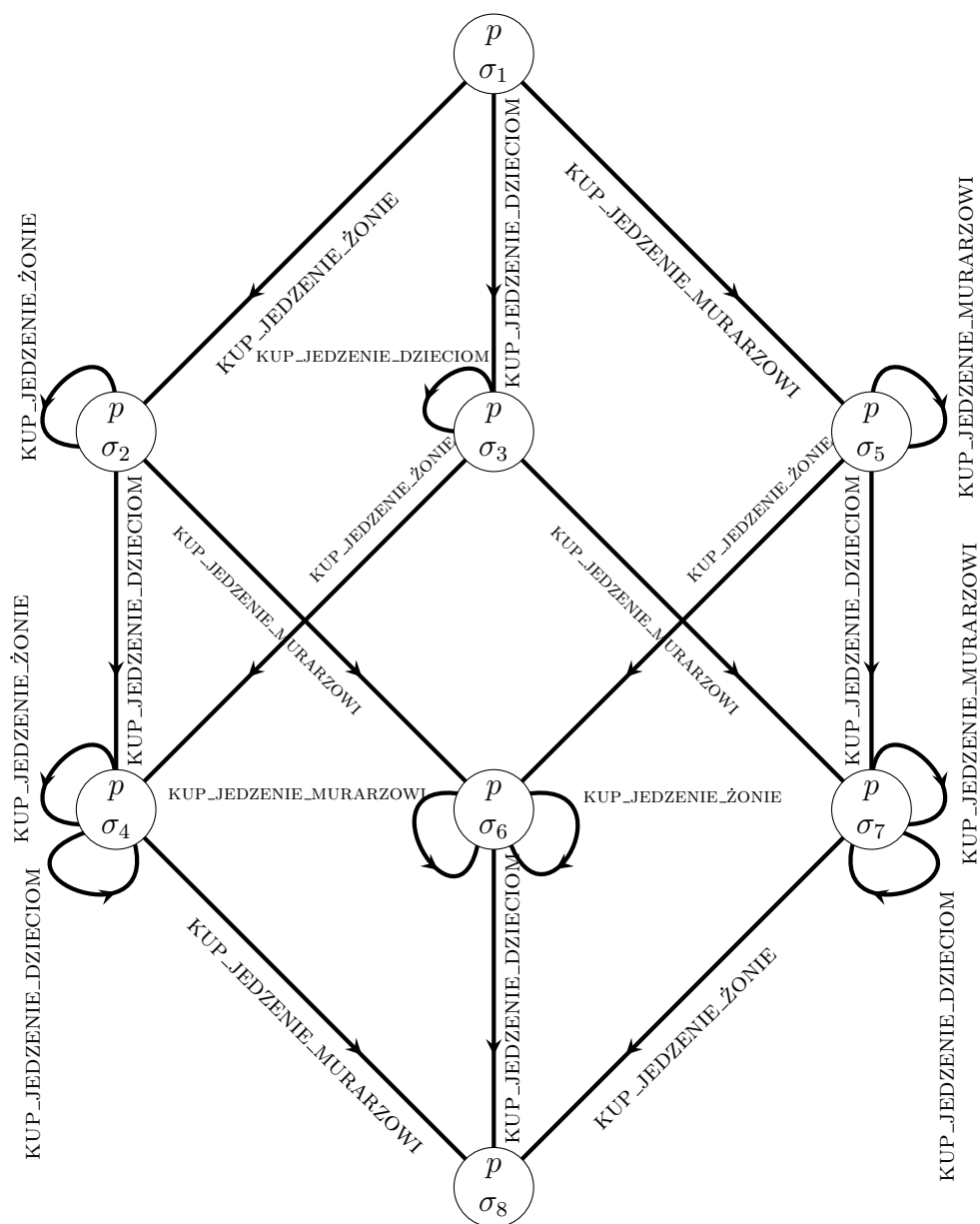
Zastanówmy się teraz nad rozwiązaniem głównego problemu naszego zadania, czyli tego czy murarz będzie w stanie wyżywić swoją rodzinę w czasie zimy. Zauważmy, że aby odpowiedzieć na to pytanie twierdząco musimy stwierdzić czy z dowolnego stanu początkowego (stany  $\sigma_1 - \sigma_4$ ) jesteśmy w stanie wykonać pewien ciąg akcji, który prowadzi nas do osiągnięcia celu którym jest stan spełniający warunek:

$$\neg \text{MurarzGłodny} \wedge \neg \text{ŻonaGłodna} \wedge \neg \text{DzieciGłodne}.$$

Nietrudno zauważyć, że jedynym stanem spełniającym ten warunek jest stan  $\sigma_8$ . Zwróćmy też uwagę, że jedynym możliwym sposobem na otrzymanie  $\sigma_8$  ze stanów  $\sigma_1 - \sigma_4$  jest wykonanie w dowolnej kolejności akcji:

- $\{\text{KUP\_JEDZENIE\_MURARZOWI}, \text{KUP\_JEDZENIE\_ŻONIE}, \text{KUP\_JEDZENIE\_DZIECIOM}\}$  w przypadku  $\sigma_1$
- $\{\text{KUP\_JEDZENIE\_MURARZOWI}, \text{KUP\_JEDZENIE\_DZIECIOM}\}$  w przypadku  $\sigma_2$
- $\{\text{KUP\_JEDZENIE\_MURARZOWI}, \text{KUP\_JEDZENIE\_ŻONIE}\}$  w przypadku  $\sigma_3$
- $\{\text{KUP\_JEDZENIE\_MURARZOWI}\}$  w przypadku  $\sigma_4$

Każda z tych akcji kosztuje 2, przy początkowym stanie zasobu (złotych monet) 5, oznacza to że jesteśmy w stanie wykonać najwyżej dwie akcje kupna jedzenia. Oznacza to, że ze stanu  $\sigma_1$  nie jesteśmy w stanie osiągnąć celu, w reszcie stanów początkowych jest to możliwe. Oznacza to, że nie możemy przyjąć, że rodzina murarza była niegłodna w czasie zimy.



Rysunek 1: Schemat pierwszego przykładu. Każda krawędź indeksowana jest nazwą akcji, przy czym istnieje tylko, jeśli zachodzi warunek:  $2 \leq p$ . Po wykonaniu każdej akcji aktualizujemy  $p$  w następujący sposób:  $p = p - 2$ .

## 5.2 Przykład 2

Kot Mruczek prowadzi bardzo systematyczny tryb życia. Głównie śpi, ale w czasie krótkich (120-minutowych) przerw między spaniem może pójść na myszy (tylko jeśli jest głodny, a po wykonaniu tej akcji przestaje być głodny). Jeśli Mruczek nie jest znudzony, to łapie mysz w ciągu 30 minut, w przeciwnym razie udaje mu się to dopiero po 90 minutach. Mruczek może również patrzeć w okno przez 10 minut z różnym skutkiem – czasami będzie po tym znudzony, a czasami nie. Mruczek może też wejść do pudełka na 30 minut (co sprawi, że nie będzie znudzony). Wiemy, że Mruczek budzi się znudzony i głodny. Czy jest szansa na to, że Mruczek tak zagospodaruje przerwę między spaniem, żeby na koniec nie być głodnym i znudzonym?

Dylemat Mruczka w naszym języku możemy zapisać tak, jak poniżej:

1. **initially** głodny, znudzony
2. **initially** 120 value
3. IDŹ\_NA\_MYSZY *causes*  $\neg$ głodny
4. IDŹ\_NA\_MYSZY *costs* 30 *if*  $\neg$ znudzony
5. IDŹ\_NA\_MYSZY *costs* 90 *if* znudzony
6. *impossible* IDŹ\_NA\_MYSZY *if*  $\neg$ głodny
7. PATRZ\_PRZEZ\_OKNO *releases* znudzony
8. PATRZ\_PRZEZ\_OKNO *costs* 10
9. WEJDŹ\_DO\_PUDEŁKA *costs* 30 *with effect*  $\neg$ znudzony

gdzie przyjmujemy:

1. zbiór fluentów:  $F = \{\text{głodny}, \text{znudzony}\}$
2. zbiór akcji:  $Ac = \{\text{IDŹ\_NA\_MYSZY}, \text{PATRZ\_PRZEZ\_OKNO}, \text{WEJDŹ\_DO\_PUDEŁKA}\}$
3. zbiór możliwych stanów zasobu:  $P = \mathbb{N}$
4. zbiory cen:  $C_{\text{IDŹ\_NA\_MYSZY}} = \{30, 90\}$ ,  $C_{\text{PATRZ\_PRZEZ\_OKNO}} = \{10\}$ ,  
 $C_{\text{WEJDŹ\_DO\_PUDEŁKA}} = \{30\}$

W naszym modelu dopuszczalnych jest  $2^2$  stanów. Stan początkowy, oznaczany jako  $\sigma_0$  jest następujący:

$$\sigma_0 = (\text{głódny}, \text{znudzony}).$$

Pozostałe stany wyglądają jak poniżej:

$$\begin{aligned}\sigma_1 &= (\neg\text{głódny}, \text{znudzony}), \\ \sigma_2 &= (\text{głódny}, \neg\text{znudzony}), \\ \sigma_3 &= (\neg\text{głódny}, \neg\text{znudzony}).\end{aligned}$$

Zobaczmy teraz, jaki stan otrzymamy po wykonaniu akcji  $\text{IDŹ\_NA\_MYSZY}$ , jeśli Murczek jest aktualnie w stanie początkowym. Liczymy najpierw  $\text{Res}_0$ :

$$\text{Res}_0(\text{IDŹ\_NA\_MYSZY}, \sigma_0, 120) = \{(\sigma_1, 30), (\sigma_3, 30)\}.$$

Następnie szukamy minimalnego zbioru  $\text{New}$  (ze względu na zawieranie):

$$\begin{aligned}\text{New}(\text{IDŹ\_NA\_MYSZY}, \sigma_0, \sigma_1) &= \{\text{głódny}\}, \\ \text{New}(\text{IDŹ\_NA\_MYSZY}, \sigma_0, \sigma_3) &= \{\text{głódny}, \text{znudzony}\}.\end{aligned}$$

Ostatecznie otrzymujemy:

$$\begin{aligned}\text{Res}_{RW}(\text{IDŹ\_NA\_MYSZY}, \sigma_0, 120) &= \{(\sigma_1, 30)\}, \\ \text{Res}(\text{IDŹ\_NA\_MYSZY}, \sigma_0, 120) &= \{(\sigma_1, 30)\}.\end{aligned}$$

Zatem po wykonaniu akcji  $\text{IDŹ\_NA\_MYSZY}$  jesteśmy w stanie  $\sigma_1$ , a wartość naszego zasobu wynosi 30. Ze stanu  $\sigma_1$  możemy już wykonać tylko akcje  $\text{WEJDŹ\_DO\_PUDEŁKA}$  i  $\text{PATRZ\_PRZEZ\_OKNO}$ , ponieważ akcja  $\text{IDŹ\_NA\_MYSZY}$  może być wykonana wyłącznie, jeśli kot jest głódny. Zastanówmy się zatem, w którym ze stanów znajdzie się Mruczek, jeśli wykona teraz akcję  $\text{WEJDŹ\_DO\_PUDEŁKA}$ .

$$\text{Res}_0(\text{WEJDŹ\_DO\_PUDEŁKA}, \sigma_1, 30) = \{(\sigma_2, 0), (\sigma_3, 0)\}.$$

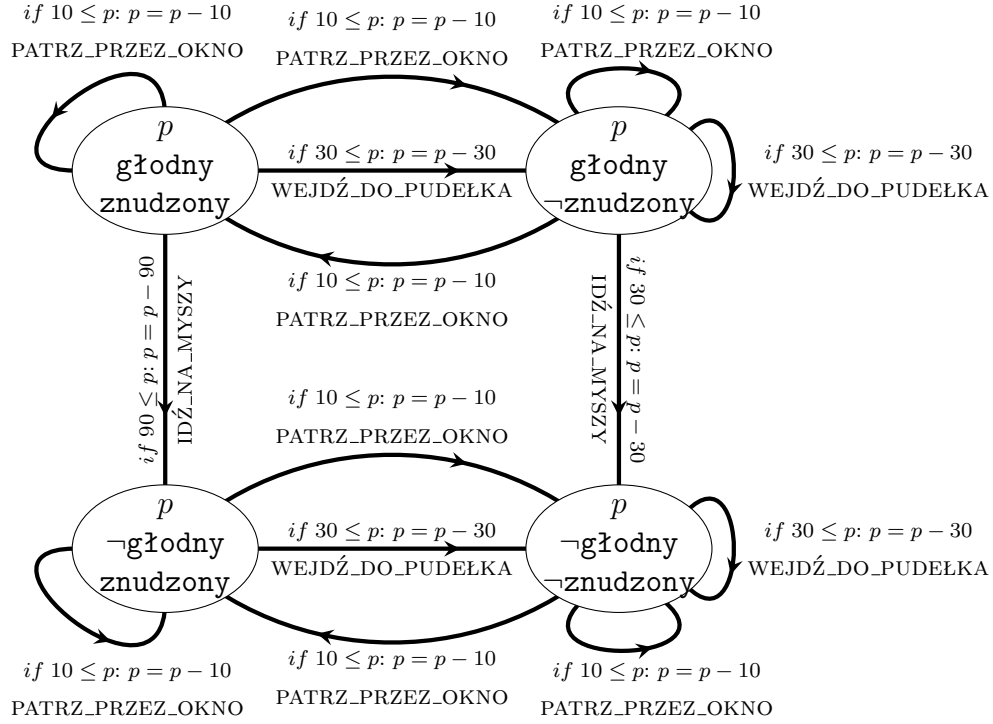
Szukamy minimalnego zbioru  $\text{New}$  (ze względu na zawieranie):

$$\begin{aligned}\text{New}(\text{WEJDŹ\_DO\_PUDEŁKA}, \sigma_1, \sigma_2) &= \{\text{głódny}, \text{znudzony}\}, \\ \text{New}(\text{WEJDŹ\_DO\_PUDEŁKA}, \sigma_1, \sigma_3) &= \{\text{znudzony}\}.\end{aligned}$$

Ostatecznie otrzymujemy:

$$\begin{aligned}\text{Res}_{RW}(\text{WEJDŹ\_DO\_PUDEŁKA}, \sigma_1, 30) &= \{(\sigma_3, 0)\}, \\ \text{Res}(\text{WEJDŹ\_DO\_PUDEŁKA}, \sigma_1, 30) &= \{(\sigma_3, 0)\}.\end{aligned}$$

Po tych dwóch akcjach stan zasobu został już całkowicie wyczerpany, ale Mruczek ostatecznie nie jest ani głódny, ani znudzony – co chcieliśmy osiągnąć.



Rysunek 2: Schemat drugiego przykładu.

### 5.3 Przykład 3

Przyjrzyjmy się następującej historii:

Zarządzamy pracą elektrowni. Zasoby oznaczają wkład pracy potrzebny do wykonania danej czynności. Wyłączanie zasilania i chłodzenia nie jest pracochłonne, dlatego nie wymaga zasobów. Włączenie zasilania lub chłodzenia, nie jest już tak łatwe i wymaga 1 zasobu. Jeśli elektrownia działa bez chłodzenia, to dochodzi do awarii. Chłodzenie nie może działać, ani być włączone jeśli zasilanie nie działa. Elektrownia ma prawie wszystkie najnowsze zabezpieczenia, dlatego nie można przerywać jej pracy bez włączonego układu chłodniczego. Włączenie lub wyłączenie elektrowni kosztuje 3 zasoby. Włączenie elektrowni wymaga aby zasilanie było włączone. Napawa elektrowni jest 'najdroższym' procesem, ponieważ kosztuje aż 10 zasobów. Naprawa odbywa się w przypadku odcięcia zasilania, co wcale nie oznacza, że elektrownia nie jest aktywna. Pytanie brzmi - czy mając 15 zasobów możemy zawsze usunąć awarię?

Podany problem można przedstawić w postaci następującego zagadnienia:

1. **initialy 15 value**

2. **initially** Awaria
3. **always**  $\neg$  Chłodzenie  $\wedge$  Aktywność  $\Rightarrow$  Awaria
4. **always**  $\neg$  Zasilanie  $\Rightarrow \neg$  Chłodzenie
5. ZASILANIE-ON *costs* 1 *with effect* Zasilanie
6. ZASILANIE-OFF *causes*  $\neg$  Zasilanie
7. CHŁODZENIE-ON *costs* 1 *if* Zasilanie *with effect* Chłodzenie
8. CHŁODZENIE-OFF *causes*  $\neg$  Chłodzenie
9. AKTYWNOŚĆ-ON *costs* 3 *if* Zasilanie *with effect* Aktywność
10. AKTYWNOŚĆ-OFF *costs* 3 *if* Chłodzenie *with effect*  $\neg$  Aktywność
11. NAPRAWA *costs* 10 *if*  $\neg$  Zasilanie *with effect*  $\neg$  Awaria

gdzie jako:

1. zbiór fluentów przyjmujemy:  $F = \{\text{Chłodzenie, Zasilanie, Aktywność, Awaria}\}$
2. zbiór akcji przyjmujemy:  $Ac = \{\text{ZASILANIE-ON, ZASILANIE-OFF, CHŁODZENIE-ON, CHŁODZENIE-OFF, AKTYWNOŚĆ-ON, AKTYWNOŚĆ-OFF, NAPRAWA}\}$
3. zbiorem możliwych stanów zasobu jest  $P = \mathbb{N}$
4. zbiory cen:
  - (a)  $C_{\text{ZASILANIE-OFF}} = C_{\text{CHŁODZENIE-OFF}} = \{0\}$ ,
  - (b)  $C_{\text{ZASILANIE-ON}} = C_{\text{CHŁODZENIE-ON}} = \{1\}$ ,
  - (c)  $C_{\text{AKTYWNOŚĆ-ON}} = C_{\text{AKTYWNOŚĆ-OFF}} = \{3\}$ ,
  - (d)  $C_{\text{NAPRAWA}} = \{10\}$

Wszystkich stanów jakie można utworzyć ze zbioru fluentów jest  $2^4$ , przy czym w naszym modelu dopuszczalnych jest tylko 10 możliwości, co jest spowodowane tym, że w zapisie naszego problemu występują ograniczenia:

**always**  $\neg$  Chłodzenie  $\wedge$  Aktywność  $\Rightarrow$  Awaria

Co wyklucza stany:

1.  $\sigma_{11} = (\neg \text{Chłodzenie}, \neg \text{Zasilanie}, \text{Aktywność}, \neg \text{Awaria})$



$$2. \sigma_{12} = (\neg \text{Chłodzenie}, \text{Zasilanie}, \text{Aktywność}, \neg \text{Awaria})$$

**always**  $\neg \text{Zasilanie} \Rightarrow \neg \text{Chłodzenie}$

Co wyklucza stany:

$$1. \sigma_{13} = (\text{Chłodzenie}, \neg \text{Zasilanie}, \text{Aktywność}, \text{Awaria})$$

$$2. \sigma_{14} = (\text{Chłodzenie}, \neg \text{Zasilanie}, \neg \text{Aktywność}, \text{Awaria})$$

$$3. \sigma_{15} = (\text{Chłodzenie}, \neg \text{Zasilanie}, \text{Aktywność}, \neg \text{Awaria})$$

$$4. \sigma_{16} = (\text{Chłodzenie}, \neg \text{Zasilanie}, \neg \text{Aktywność}, \neg \text{Awaria})$$

Zastanówmy się nad stanami początkowymi. Są to stany:

$$1. \sigma_1 = (\neg \text{Chłodzenie}, \neg \text{Zasilanie}, \neg \text{Aktywność}, \text{Awaria})$$

$$2. \sigma_2 = (\neg \text{Chłodzenie}, \neg \text{Zasilanie}, \text{Aktywność}, \text{Awaria})$$

$$3. \sigma_3 = (\neg \text{Chłodzenie}, \text{Zasilanie}, \neg \text{Aktywność}, \text{Awaria})$$

$$4. \sigma_4 = (\neg \text{Chłodzenie}, \text{Zasilanie}, \text{Aktywność}, \text{Awaria})$$

$$5. \sigma_5 = (\text{Chłodzenie}, \text{Zasilanie}, \neg \text{Aktywność}, \text{Awaria})$$

$$6. \sigma_6 = (\text{Chłodzenie}, \text{Zasilanie}, \text{Aktywność}, \text{Awaria})$$

wynika to z faktu, że w naszym programie występuje wyrażenie: **initially Awaria**.

Resztę stanów (stany nie będące stanami początkowymi i należące do zbioru wszystkich stanów dopuszczalnych  $\Sigma$ ) oznaczmy jako:

$$1. \sigma_7 = (\neg \text{Chłodzenie}, \neg \text{Zasilanie}, \neg \text{Aktywność}, \neg \text{Awaria})$$

$$2. \sigma_8 = (\neg \text{Chłodzenie}, \text{Zasilanie}, \neg \text{Aktywność}, \neg \text{Awaria})$$

$$3. \sigma_9 = (\text{Chłodzenie}, \text{Zasilanie}, \neg \text{Aktywność}, \neg \text{Awaria})$$

$$4. \sigma_{10} = (\text{Chłodzenie}, \text{Zasilanie}, \text{Aktywność}, \neg \text{Awaria})$$

Naszym celem jest próba przeprowadzenia stanów od  $\sigma_1$  do  $\sigma_6$ , czyli stanów początkowych, w stany od  $\sigma_7$  do  $\sigma_{10}$  z wykorzystaniem maksymalnie 15 zasobów. Pokażemy, że wykonanie sekwencji akcji ZASILANIE-ON, CHŁODZENIE-ON, AKTYWNOŚĆ-OFF, ZASILANIE-OFF, NAPRAWA, których łączny koszt wynosi  $C_{\text{ZASILANIE-ON}} + C_{\text{CHŁODZENIE-ON}} + C_{\text{AKTYWNOŚĆ-OFF}} + C_{\text{ZASILANIE-OFF}} + C_{\text{NAPRAWA}} = 1 + 1 + 3 + 0 + 10 = 15$  zawsze zaprowadzi nas, bez naruszania ograniczeń, do stanu, w którym fluent Awaria jest niezaprzeczony. Zaczniemy od wykonania na stanach od  $\sigma_1$  do  $\sigma_6$  akcji ZASILANIE-ON. Po obliczeniach otrzymujemy:

- $Res(ZASILANIE-ON, \sigma_1, 15) = \{(\sigma_3, 14)\}$
- $Res(ZASILANIE-ON, \sigma_2, 15) = \{(\sigma_4, 14)\}$
- $Res(ZASILANIE-ON, \sigma_3, 15) = \{(\sigma_3, 14)\}$
- $Res(ZASILANIE-ON, \sigma_4, 15) = \{(\sigma_4, 14)\}$
- $Res(ZASILANIE-ON, \sigma_5, 15) = \{(\sigma_5, 14)\}$
- $Res(ZASILANIE-ON, \sigma_6, 15) = \{(\sigma_6, 14)\}$

Widzimy, że dla stanów od  $\sigma_3$  do  $\sigma_6$  wykonanie akcji ZASILANIE-ON nie jest konieczne, ponieważ stany nie zmieniają się. Jeśli pokażemy, że ze stanu  $\sigma_x$  możemy dojść do stanu  $\sigma_y$  z wykorzystaniem  $k$  zasobów, to możemy również przejść ze stanu  $\sigma_x$  do stanu  $\sigma_y$  posiadając  $k+i$  zasobów, gdzie  $i \in \{1, 2, 3, \dots\}$ . Oczywiście przy założeniu, że wykonanie akcji nie może wymagać ujemnej liczby zasobów (co zakładamy). W dalszej części rozumowania nie musimy więc pokazywać, że stany  $\sigma_3$  oraz  $\sigma_4$  można przeprowadzić w stany od  $\sigma_7$  do  $\sigma_{10}$  posiadając 15 zasobów, jeśli pokażemy, że można to zrobić z wykorzystaniem 14 zasobów. Wykonajmy teraz akcję CHŁODZENIE-ON.

- $Res(CHŁODZENIE-ON, \sigma_3, 14) = \{(\sigma_5, 13)\}$
- $Res(CHŁODZENIE-ON, \sigma_4, 14) = \{(\sigma_6, 13)\}$
- $Res(CHŁODZENIE-ON, \sigma_5, 15) = \{(\sigma_5, 14)\}$
- $Res(CHŁODZENIE-ON, \sigma_6, 15) = \{(\sigma_6, 14)\}$

Widać, że dla stanów od  $\sigma_5$  oraz  $\sigma_6$  wykonanie akcji CHŁODZENIE-ON nie jest konieczne. Wykonując akcję CHŁODZENIE-ON przeprowadziliśmy stany  $\sigma_3$  i  $\sigma_4$  w stany  $\sigma_5$  i  $\sigma_6$ . W dalszych rozważaniach będziemy się więc zajmować już tylko stanami  $\sigma_5$  oraz  $\sigma_6$ . Wykonajmy akcję AKTYWNOSC-OFF.

- $Res(AKTYWNOSC-OFF, \sigma_5, 13) = \{(\sigma_5, 10)\}$
- $Res(AKTYWNOSC-OFF, \sigma_6, 13) = \{(\sigma_5, 11)\}$

Doszliśmy do sytuacji, w której rozpatrujemy już tylko jeden stan  $\sigma_5$ . Jeśli pokażemy, że możemy go przeprowadzić w któryś ze stanów od  $\sigma_7$  do  $\sigma_{10}$  posiadając 10 zasobów to udowodnimy, że wszystkie stany początkowe można przeprowadzić w stan, w którym fluent Awaria jest niezaprzeczony, z wykorzystaniem 15 zasobów. Wykonajmy akcję ZASILANIE-OFF. Zauważmy, że spowoduje ona zmianę nie tylko fluentu Zasilanie ale również Chłodzenie. Stanie się tak z powodu ograniczenia **always**  $\neg$  Zasilanie  $\Rightarrow$   $\neg$  Chłodzenie.

- $Res(ZASILANIE-OFF, \sigma_5, 10) = \{(\sigma_1, 10)\}$

Ostatnią wykonaną akcją będzie NAPRAWA.

- $Res(ZASILANIE-OFF, \sigma_1, 10) = \{(\sigma_7, 0)\}$

Co kończy dowód. Pokazaliśmy, że z wykorzystaniem 15 zasobów rzeczywiście można z każdego stanu początkowego dojść do stanu od  $\sigma_7$  do  $\sigma_{10}$ . Wyniki postępowania możnaby przedstawić na grafie o 10 wierzchołkach i krawędziach indeksowanych nazwami akcjami oraz ich ceną. Przejścia, które znaleźliśmy nie zawsze są optymalne (np. ze stanu  $\sigma_1$  można było od razu przejść do stanu  $\sigma_7$ , zamiast do  $\sigma_3$ ), ale spełniają wszystkie postanowione ograniczenia. Odpowiedź na postawione w zadaniu pytanie brzmi więc, że mając 15 zasobów możemy zawsze usunąć awarię.

## 5.4 Przykład 4

Marysia mieszka w Bartoszycach. W przyszłą sobotę wybiera się do swojej szkoły na bal karnawałowy (również w Bartoszycach). Jednak zanim wybierze się na ten bal, zauważyła, że nie ma się w co ubrać. Marysia posiada 100 zł. W najbliższym sklepie "Bolek", Marysia może kupić sukienkę za 60 zł oraz buty za 50 zł. Może również wybrać się do sąsiedniego miasta Olsztyn, do którego jeździ autobus. Bilet do Olsztyna kosztuje 10 zł. Wieczorem Marysia powinna wrócić tym samym autobusem do domu (koszt również wynosi 10 zł). W znajdującym się w Olsztynie sklepie "Tola" może kupić sukienkę za 30 zł, oraz buty za 80 zł. Czy Marysi starczy pieniędzy na potrzebne zakupy?

Problem można przedstawić do następującego zagadnienia:

1. **initially** 100 value
2. **initially**  $\neg$  PosiadaSukienkę
3. **initially**  $\neg$  PosiadaButy
4. **initially** WBartoszykach
5. **AUTOBUSOLSZTYN** costs 10 if WBartoszykach with effect  $\neg$  WBartoszykach
6. **AUTOBUSBARTOSZYCE** costs 10 if  $\neg$  WBartoszykach with effect WBartoszykach
7. **KUPBUTYBOLEK** costs 50 if WBartoszykach with effect PosiadaButy
8. **KUPBUTYTOLA** costs 80 if  $\neg$  WBartoszykach with effect PosiadaButy

9. KUPSUKIENKĘBOLEK *costs* 60 *if* WBartoszczach *with effect* PosiadaSukienkę
10. KUPSUKIENKĘTOLA *costs* 30 *if*  $\neg$  WBartoszczach *with effect* PosiadaSukienkę

gdzie:

1. zbiór fluentów przyjmujemy:  $F = \{\text{PosiadaSukienkę}, \text{PosiadaButy}, \text{WBartoszczach}\}$
2. zbiór akcji przyjmujemy:  $Ac = \{\text{AUTOBUSOLSZTYN}, \text{AUTOBUS-BARTOSZYCE}, \text{KUPBUTYBOLEK}, \text{KUPBUTYTOLA}, \text{KUPSUKIENKĘBOLEK}, \text{KUPSUKIENKĘTOLA}\}$
3. zbiorem możliwych stanów zasobu jest  $P = \mathbb{N}$
4. zbiory cen:
  - (a)  $C_{\text{AUTOBUSOLSZTYN}} = C_{\text{AUTOBUSBARTOSZYCE}} = \{10\}$ ,
  - (b)  $C_{\text{KUPSUKIENKETOLA}} = \{30\}$ ,
  - (c)  $C_{\text{KUPBUTYBOLEK}} = \{50\}$ ,
  - (d)  $C_{\text{KUPSUKIENKEBOLEK}} = \{60\}$ ,
  - (e)  $C_{\text{KUPBUTYTOLA}} = \{80\}$

Wszystkich stanów jakie można utworzyć ze zbioru fluentów jest  $2^3$  Istnieje tylko jeden stan początkowy:

- $\sigma_1 = (\neg \text{PosiadaSukienkę}, \neg \text{PosiadaButy}, \text{WBartoszczach})$

Resztę stanów (stany nie będące stanami początkowymi i należące do zbioru wszystkich stanów dopuszczalnych  $\Sigma$ ) oznaczmy jako:

- $\sigma_2 = (\text{PosiadaSukienkę}, \text{PosiadaButy}, \text{WBartoszczach})$
- $\sigma_3 = (\text{PosiadaSukienkę}, \text{PosiadaButy}, \neg \text{WBartoszczach})$
- $\sigma_4 = (\text{PosiadaSukienkę}, \neg \text{PosiadaButy}, \text{WBartoszczach})$
- $\sigma_5 = (\text{PosiadaSukienkę}, \neg \text{PosiadaButy}, \neg \text{WBartoszczach})$
- $\sigma_6 = (\neg \text{PosiadaSukienkę}, \text{PosiadaButy}, \text{WBartoszczach})$
- $\sigma_7 = (\neg \text{PosiadaSukienkę}, \text{PosiadaButy}, \neg \text{WBartoszczach})$

- $\sigma_8 = (\neg \text{PosiadaSukienkę}, \neg \text{PosiadaButy}, \neg \text{WBartoszczach})$

Naszym celem jest próba przeprowadzenia stanu początkowego  $\sigma_1$  do stanu  $\sigma_2$ . Pokażemy, że wykonanie sekwencji akcji AUTOBUSOLSZTYN, KUPSUKIENKĘTOLA, AUTOBUSBARTOSZYCE, KUPBUTYBOLEK, których łączny koszt wynosi  $C_{\text{AUTOBUSOLSZTYN}} + C_{\text{KUPSUKIENKĘTOLA}} + C_{\text{AUTOBUSBARTOSZYCE}} + C_{\text{KUPBUTYBOLEK}} = 10 + 30 + 10 + 50 = 100$ , doprowadzi to takiego przejścia. Wykonajmy kolejne obliczenia:

1.  $\text{Res}(\text{AUTOBUSOLSZTYN}, \sigma_1, 100) = \{(\sigma_8, 90)\}$
2.  $\text{Res}(\text{KUPSUKIENKĘTOLA}, \sigma_8, 90) = \{(\sigma_5, 60)\}$
3.  $\text{Res}(\text{AUTOBUSBARTOSZYCE}, \sigma_5, 60) = \{(\sigma_4, 50)\}$
4.  $\text{Res}(\text{KUPBUTYBOLEK}, \sigma_4, 50) = \{(\sigma_2, 0)\}$

Uwaga - nie jest to jedyna sekwencja akcji o takim skutku. Wykonanie akcji KUPBUTYBOLEK, AUTOBUSOLSZTYN, KUPSUKIENKĘTOLA, AUTOBUSBARTOSZYCE również stanowi poprawne rozwiązanie problemu. Kolejne przejścia wyglądają wtedy następująco:

1.  $\text{Res}(\text{KUPBUTYBOLEK}, \sigma_1, 100) = \{(\sigma_6, 50)\}$
2.  $\text{Res}(\text{AUTOBUSOLSZTYN}, \sigma_6, 50) = \{(\sigma_7, 40)\}$
3.  $\text{Res}(\text{KUPSUKIENKĘTOLA}, \sigma_7, 40) = \{(\sigma_3, 10)\}$
4.  $\text{Res}(\text{AUTOBUSBARTOSZYCE}, \sigma_3, 10) = \{(\sigma_2, 0)\}$

W ten sposób pokazaliśmy, że z wykorzystaniem 100zł rzeczywiście można ze stanu początkowego  $\sigma_1$  dojść do stanu  $\sigma_2$ . Wyniki postępowania można przedstawić na grafie o wierzchołkach indeksowanych stanami i krawędziach indeksowanych nazwami akcjami oraz ich ceną. Odpowiedź na postawione w zadaniu pytanie brzmi więc, że Marysi starczy pieniędzy na wydatki związane z balem.

## 5.5 Przykład 5

Robert jest nastolatkiem, który przepada za grą w piłkę i często spotyka się z gronem swoich znajomych. Pewnego dnia obudził się wypoczęty (30 jednostek) i snuł swoje plany odnośnie kolejnych meczów i spotkań towarzyskich. Każdy mecz wymaga, by Robert był sprawny, jest bardzo męczący (10 jednostek) i powoduje liczne kontuzje, tak że tylko maść Altacet jest w stanie przywrócić sprawność Roberta po meczu. Zastosowanie maści Altacet

to drobnostka, więc koszt aplikowania Altacetu to 0 jednostek. Dodatkowo, humor Roberta zmienia się zależnie od przebiegu meczu: raz Robert jest zadowolony po meczu, innym razem nie.

Spotkania towarzyskie natomiast nie kosztują Roberta wiele wysiłku, szczególnie, gdy jest zadowolony - wówczas koszt to 1 jednostka. Gdy Robert nie jest zadowolony, uczestniczenie w spotkaniu jest dla niego nieco męczące - 5 jednostek. Co więcej, spotkania towarzyskie mają różny efekt, w zależności od tego, w jakim humorze jest Robert. Gdy jest zadowolony, budzi wielką sympatię i po spotkaniu cieszy się popularnością - jest lubiany. Natomiast gdy humor mu nie dopisuje, zwykle zachowuje się kapryśnie i docina innym, tak że po spotkaniu nie jest lubiany. Jaki jest minimalny koszt akcji, które sprawią że Robert będzie lubiany jeśli obudził się niezadowolony i nie lubiany?

Ten problem można przedstawić następująco w naszym języku:

1. **initially**  $\neg$  Zadowolony
2. **initially**  $\neg$  Lubiany
3. **initially** 30 value
4. MECZ *costs* 10 **if** Sprawny *with effect*  $\neg$  Sprawny
5. MECZ **releases** Zadowolony
6. ALTACET *costs* 0 *with effect* Sprawny
7. SPOTKANIE *costs* 1 **if** Zadowolony *with effect* Lubiany
8. SPOTKANIE *costs* 5 **if**  $\neg$  Zadowolony *with effect*  $\neg$  Lubiany

gdzie jako:

1. zbiór fluentów przyjmujemy:  $F = \{ \text{Zadowolony, Sprawny, Lubiany} \}$
2. zbiór akcji przyjmujemy:  $Ac = \{ \text{MECZ, ALTACET, SPOTKANIE} \}$
3. zbiorem możliwych stanów zasobu jest  $P = \mathbb{N}$
4. zbiory cen:
  - (a)  $C_{ALTACET} = \{0\}$ ,
  - (b)  $C_{SPOTKANIE} = \{1, 5\}$ ,
  - (c)  $C_{MECZ} = \{10\}$ ,

Wszystkich możliwych stanów fluentów jest  $2^3$ . Stanów początkowych jest  $2^1$ . Są to stany:

1.  $\sigma_1 = (\text{Sprawny}, \neg \text{Zadowolony}, \neg \text{Lubiany})$
2.  $\sigma_2 = (\neg \text{Sprawny}, \neg \text{Zadowolony}, \neg \text{Lubiany})$

wynika to z faktu, że w naszym programie występują wyrażenia:

1. **initially**  $\neg$  Zadowolony
2. **initially**  $\neg$  Lubiany

Stany należące do zbioru wszystkich stanów dopuszczalnych i nie będące stanami początkowymi oznaczmy jako:

1.  $\sigma_3 = (\text{Sprawny}, \text{Zadowolony}, \text{Lubiany})$
2.  $\sigma_4 = (\text{Sprawny}, \text{Zadowolony}, \neg \text{Lubiany})$
3.  $\sigma_5 = (\text{Sprawny}, \neg \text{Zadowolony}, \text{Lubiany})$
4.  $\sigma_6 = (\neg \text{Sprawny}, \text{Zadowolony}, \text{Lubiany})$
5.  $\sigma_7 = (\neg \text{Sprawny}, \text{Zadowolony}, \neg \text{Lubiany})$
6.  $\sigma_8 = (\neg \text{Sprawny}, \neg \text{Zadowolony}, \text{Lubiany})$

Naszym celem jest prowadzenie stanów  $\sigma_1$  i  $\sigma_2$ , czyli stanów początkowych, w stany  $\sigma_3$ ,  $\sigma_5$ ,  $\sigma_6$  lub  $\sigma_8$ , które spełniają warunek, że fluent Lubiany jest niezaprzeczony z wykorzystaniem minimalnej ilości zasobów. Przeanalizujemy wszystkie możliwe akcje, czyli MECZ, ALTACET oraz SPOTKANIE dla stanów  $\sigma_1$  i  $\sigma_2$ . Załóżmy, że początkowa ilość jednostek wynosi  $x$ . Chcemy znaleźć taki ciąg akcji (osobny dla  $\sigma_1$  i osobny dla  $\sigma_2$ ), który doprowadzi do stanu w którym Robert jest Lubiany, a wartość  $x$  zmniejszy się o najmniejszą liczbę. Dla stanu  $\sigma_1$ :

- $Res(\text{MECZ}, \sigma_1, x) = \{(\sigma_2, x - 10), (\sigma_7, x - 10)\}$
- $Res(\text{ALTACET}, \sigma_1, x) = \{(\sigma_1, x)\}$
- $Res(\text{SPOTKANIE}, \sigma_1, x) = \{(\sigma_1, x - 5)\}$

Dla stanu  $\sigma_2$ :

- w stanie  $\sigma_2$  nie można wykonać akcji MECZ, ponieważ fluent Sprawny jest zaprzeczony.

- $Res(ALTACET, \sigma_2, x) = \{(\sigma_1, x)\}$
- $Res(SPOTKANIE, \sigma_2, x) = \{(\sigma_2, x - 5)\}$

Ze stanu  $\sigma_2$  możemy przejść do stanów  $\sigma_1$  oraz  $\sigma_2$ . Z kolei, ze stanu  $\sigma_1$  możemy przejść do stanów  $\sigma_1$ ,  $\sigma_2$  oraz  $\sigma_7$ . Może się tak jednak zdarzyć, że ze stanu  $\sigma_1$  będziemy przechodzili wyłącznie w stany  $\sigma_1$  oraz  $\sigma_2$ . W pesymistycznym przypadku, nigdy nie wyjdziemy poza stany  $\sigma_1$  oraz  $\sigma_2$ . Prowadzi to do konkluzji, że nie istnieje ciąg akcji, który zagwarantuje, osiągnięcie któregoś ze stanów w którym fluent *Lubiany* jest niezaprzeczony (stany  $\sigma_3$ ,  $\sigma_5$ ,  $\sigma_6$ ,  $\sigma_8$ ). Skoro nie istnieje taki ciąg akcji, to nie można podać jego minimalnego kosztu.



## 6 Wygląd programu

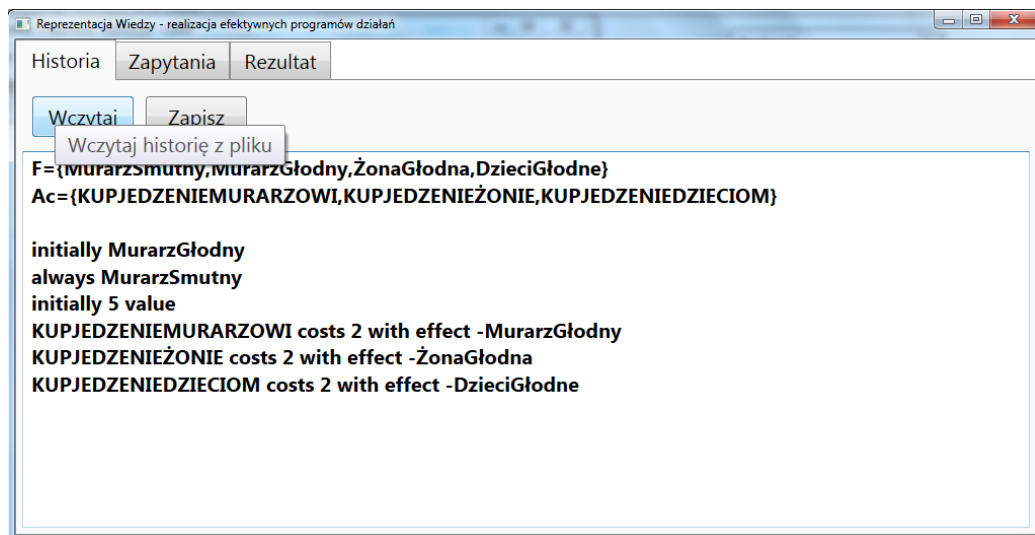
Program, który został przygotowany przez nas będzie umożliwiał obliczanie zapytań, które zostały zdefiniowane w treści zadania. Użytkownik będzie miał do dyspozycji trzy zakładki, które będą wykorzystane do przeprowadzenia tych obliczeń:

- Historia
- Zapytania
- Rezultat

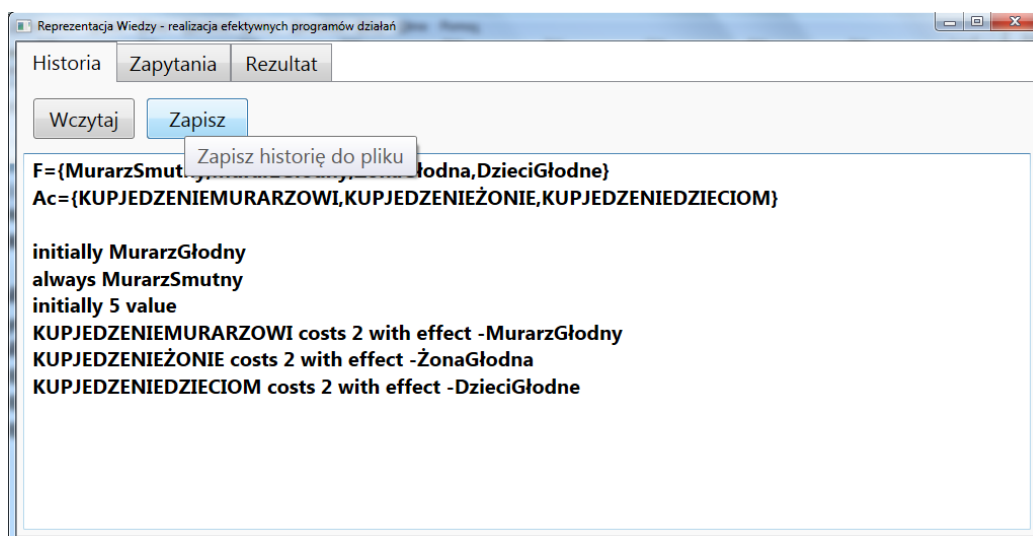
Każda z nich będzie udostępniała fragment funkcjonalności.

### 6.1 Zakładka *Historia*

W tej zakładce Użytkownik może wykonać kilka akcji - przede wszystkim, może wczytać plik z zapisanymi danymi wejściowymi w ustalonym przez nas formacie. Po naciśnięciu przycisku "Wczytaj", otworzone zostanie okno dialogowe, w którym możemy wybrać katalog i plik w rozszerzeniu .txt. Po wykonaniu tej operacji, zawartość pliku pojawi się nam na ekranie.



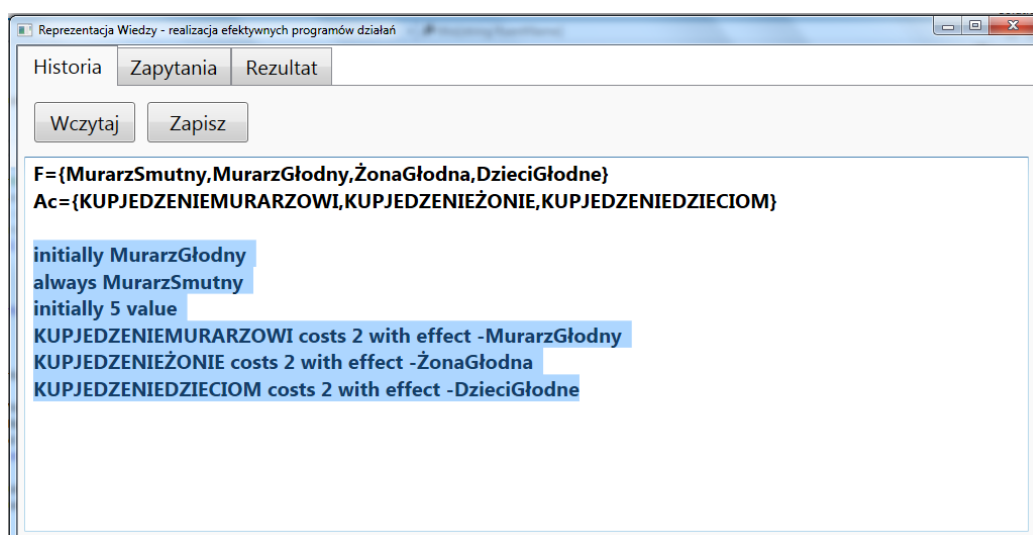
Oczywiście możemy też wykonać przeciwną operację - zapis do pliku .txt - wciskając przycisk "Zapisz", przeniesiemy się do okna dialogowego, w którym będziemy mogli wybrać gdzie i pod jaką nazwą zapisana zostanie zawartość okna.



Ponadto, użytkownik może sam ręcznie wpisać historię w oknie dialogowym. W tym celu, musi stworzyć poniższą strukturę:

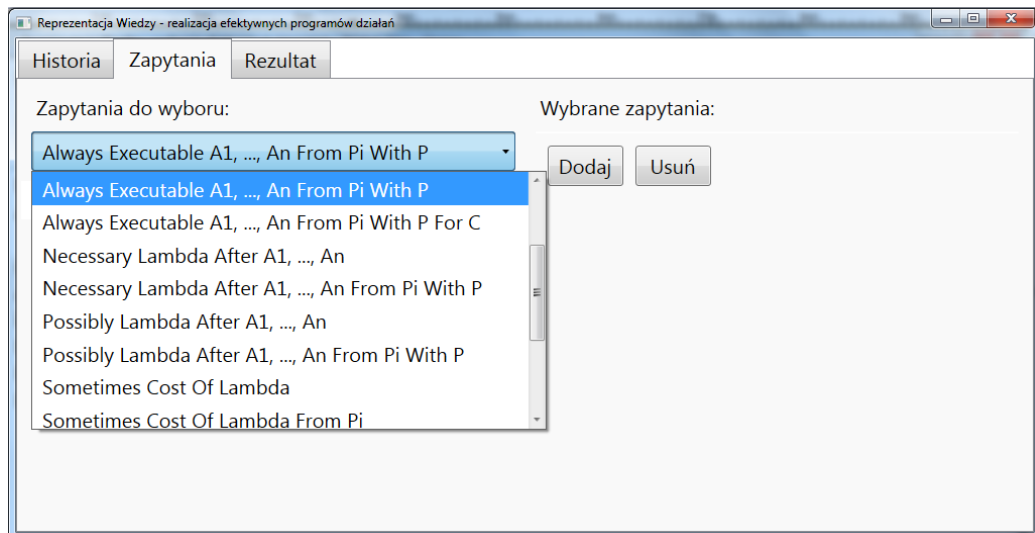
Pierwszy jego wiersz to zbiór fluentów. Fluenty oddzielone są między sobą przecinkami. Analogicznie, drugi wiersz zawiera zbiór akcji dostępnych do wykonania w programie. Trzeci wiersz – linia odstępu (dla zwiększenia czytelności pliku). Od czwartego wiersza zaczynają się inicjujące program instrukcje.

Przykład poniżej obrazuje, jak powinna wyglądać struktura i edycja historii w programie:



## 6.2 Zakładka *Zapytania*

W tej zakładce będziemy mogli ustalić kwerendy, na które będzie odpowiadał nasz program. Te kwerendy możemy wybrać z listy wszystkich dostępnych kwerend (dostępna w zakładce pod nazwą „Zapytania do wyboru”).



Po wybraniu odpowiadającej nam kwerendy, musimy doprecyzować parametry w niej określone. Stąd poniżej znajduje się okno do wpisania wartości odpowiednich parametrów.

Dodatkowo, w oknie określania parametrów znajduje się przycisk „Dodaj następne  $A_n$ ”, który pozwoli na dodanie kolejnych pól na akcje  $A_2, A_3, A_4, \dots, A_n$ . Jedno kliknięcie przycisku, powoduje pojawienie się kolejnego przycisku  $A_x$  gdzie  $A_{x-1}$  to ostatnie wyświetlane w oknie pole. Analogicznie można zmniejszyć liczbę pól akcji.

Jeśli pole pozostaje wyszarzone (nieaktywne), oznacza to, że dany symbol nie występuje w naszej kwerendzie. Jeśli pole jest aktywne, w momencie kiedy obsługujący program będzie chciał dodać kwerendę, powinien zostać wyświetlony komunikat, że nie wszystkie parametry zostały uzupełnione.

Po wypełnieniu parametru, użytkownik może wcisnąć przycisk „Dodaj”, co powoduje, że wypełniona kwerenda trafi na listę „Wybrane zapytania”.

W każdym momencie użytkownik może usunąć z listy „Wybrane zapytania” kwerendę. Wystarczy ją nacisnąć i wcisnąć przycisk „Usuń”.

### 6.3 Zakładka *Rezultat*

W tej zakładce Użytkownik może uzyskać odpowiedzi na wcześniej zadane przez nas pytania (kwerendy) określone w zakładce "Zapytania". Odpowiedzią możliwą jest True/False. Informuje nas to, czy kwerenda jest wykonalna(True), czy też nie (False).

