Wnioskowanie o działaniach z wykorzystaniem semantyki Dijkstry dla języków programowania: podejście imperatywne

Ewa Madalińska

na podstawie pracy:

W. Łukaszewicz, E. Madalińska-Bugaj. Reasoning about Action and Change Using Dijkstra's Semantics for Programming Languages: Preliminary Report. *Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995, p. 1950-1955.

Wnioskowanie o scenariuszach

Definicja 1
Scenariuszem nazywamy wyrażenie

$$SC = \langle\alpha_0\rangle\ A_1; \langle\alpha_1\rangle\ A_2; \ldots; \langle\alpha_{n-1}\rangle\ A_n\ \langle\alpha_n\rangle \quad (1)$$

gdzie $A_1, \ldots, A_n$ są symbolami **akcji**, a $\alpha_0, \ldots, \alpha_n$ są **obserwacjami**. ▪

Intuicyjna interpretacja scenariusza jest następująca: α_0 zachodzi w stanie początkowym, następnie zostają wykonane sekwencyjnie akcje $A_1, \ldots, A_n$ oraz dla każdego $0 < i \leq n$, po wykonaniu akcji A_i zaobserwowano, że zachodzi α_i.

Opis działania:

$\{$*warunek wstępny*$_1\}$ **Nazwa** $\{$*efekt*$_1\}$
⋮
$\{$*warunek wstępny*$_n\}$ **Nazwa** $\{$*efekt*$_n\}$

Zadanie wnioskowania o scenariuszach: "Dla danego scenariusza (1), zdania γ i liczby całkowitej k określ, czy γ zachodzi po wykonaniu akcji $A_1, \ldots, A_k$".

Realizacja procesu wnioskowania o scenariuszach wymaga:

1. zdefiniowania języka, który umożliwiałby formalny opis zarówno dopuszczalnych działań, jak i stanów systemu.

2. skonstruowania mechanizmu wnioskowania o efektach działań.

3. zapewnienia pełności i poprawności skonstruowanego mechanizmu.

Przyjmujemy:

1. Jako bazę logiczną – logikę zdań. Stałe zdaniowe występujące w języku logiki będziemy nazywać fluentami.

2. Język opisu działań – język programowania Dijkstry.

3. Semantykę działań będziemy wyrażać za pomocą transformatorów formuł Dijkstry dla instrukcji reprezentującej działanie.

Przykład 1 (Yale Shooting Scenario)

$YSS = \langle a \wedge \neg l \rangle load; wait; shoot \langle T \rangle,$

gdzie:

1. **Fluenty:**
   a – oznacza, że indyk żyje;
   l – strzelba jest naładowana.

2. **Działania:**
   $load$: strzelba zostaje naładowana;
   $wait$: nic się nie dzieje;
   $shoot$: jeśli strzelba jest naładowana, to indyk zostaje zabity, w pp. brak działania.

Język definiowania działań i semantyka Dijkstry

Założenia: tylko jeden typ zmiennych – zmienne logiczne (Boolean).

V – zbiór zmiennych logicznych.

Stan nad zbiorem V – funkcja σ określona na zbiorze V o wartościach w zbiorze $\{0, 1\}$

$$\sigma : V \to \{0, 1\}$$

Język asercji $\mathcal{L}(V)$ nad zbiorem zmiennych logicznych V – zbiór formuł skonstruowanych w zwykły sposób z elementów zbioru V, spójników logicznych $(\neg, \Rightarrow, \wedge, \vee, \equiv)$ i stałych logicznych T (prawda) i F (fałsz).

β – formuła,
$\alpha_1, \ldots, \alpha_n$ – wyrażenia boolowskie,
$x_1, \ldots, x_n$ – zmienne

$\beta[x_1 \leftarrow \alpha_1, \ldots, x_n \leftarrow \alpha_n]$ oznacza formułę otrzymaną z β w wyniku jednoczesnego zastąpienia wszystkich wolnych wystąpień $x_1, \ldots, x_n$ odpowiednio przez $\alpha_1, \ldots, \alpha_n$.

Kwantyfikatory $(\forall, \exists)^*$:

$\forall x.\alpha \stackrel{\text{def}}{=} \alpha[x \leftarrow T] \wedge \alpha[x \leftarrow F],$

$\exists x.\alpha \stackrel{\text{def}}{=} \alpha[x \leftarrow T] \vee \alpha[x \leftarrow F].$

*Zauważmy, że kwantyfikatory mogą być zastosowane jedynie do zmiennych logicznych.

Wartość formuły $\alpha \in \mathcal{L}(\mathcal{V})$ *w stanie* σ, $\| \alpha \|_\sigma$ jest to element zbioru $\{0, 1\}$ wyznaczony według reguł klasycznej logiki zdań.

Stan σ *spełnia* formułę α wtw $\| \alpha \|_\sigma = 1$. Jeżeli σ spełnia α, to σ nazywamy *modelem* α. Mówiąc, że σ spełnia α, będziemy zawsze zakładać, że σ określa interpretację dla wszystkich zmiennych boolowskich występujących w α.

Transformatory formuł:

Rozważamy 3 transformatory formuł:
najsłabszy warunek wstępny – wp,
najsłabszy liberalny warunek wstępny – wlp oraz
najmocniejszy warunek końcowy – sp.

Dla dowolnej instrukcji S i formuły α:

- $wp(S, \alpha)$ jest formułą, której modelami są dokładnie wszystkie stany takie, że wykonanie S rozpoczęte w którymkolwiek z nich zakończy się w stanie spełniającym warunek α.

- $wlp(S, \alpha)$ jest formułą, której modelami są dokładnie wszystkie stany takie, że jeśli wykonanie S rozpocznie się w którymkolwiek z nich i zakończy, to stan końcowy spełnia warunek α.

- $sp(S, \alpha)$ jest formułą, której modelami są dokładnie te stany, które są osiągalne w wyniku wykonania S, jeśli S się skończy, gdy S rozpocznie się w stanie spełniającym warunek α.

$$\{\alpha\}S\{\beta\}$$

$$\begin{array}{rcl} \{wlp(\beta)\}S\{\beta\} & \text{i} & \alpha \Rightarrow wlp(\beta) \\ \{wp(\beta\}S\{\beta\} & \text{i} & \alpha \Rightarrow wp(\beta) \\ \{\alpha\}S\{sp(\alpha)\} & \text{i} & sp(\beta) \Rightarrow \alpha \end{array}$$

Semantyka instrukcji

Lista instrukcji:

skip, przypisanie, sekwencyjne złożenie, alternatywa (niedeterministyczna).

1. **Instrukcja skip.** Jest to instrukcja "pusta", tzn. jej wykonanie nie zmienia stanu obliczeń.

   $$wlp(skip, \alpha) = sp(skip, \alpha) = \alpha.$$

2. **Instrukcja przypisania.** Jest postaci $x := e$, gdzie x jest zmienną boolowską, a e jest wyrażeniem boolowskim. W wyniku wykonania instrukcji wartość x zostaje zastąpiona przez wartość wyrażenia e.

   $$wlp(x := e, \alpha) = \alpha[x \leftarrow e].$$

$$sp(x := e, \alpha) = \exists y.((x \equiv e[x \leftarrow y]) \wedge \alpha[x \leftarrow y]), \quad (2)$$

gdzie y jest nową zmienną.

Jeśli zmienna x nie występuje w wyrażeniu e, równanie (2) można uprościć do postaci

$$sp(x := e, \alpha) = (x \equiv e) \wedge \exists x.\alpha$$

a biorąc pod uwagę, że zmienne są tylko typu boolowskiego, jest to równoważne

$$sp(x := e, \alpha) = (x \equiv e) \wedge (\alpha[x \leftarrow T] \vee \alpha[x \leftarrow F]). \quad (3)$$

W przykładach często będziemy mieć do czynienia z przypisaniami $x := e$, gdzie e jest T lub F.

Wtedy (3) można zastąpić przez

$$sp(x := T, \alpha) = x \wedge (\alpha[x \leftarrow T] \vee \alpha[x \leftarrow F])$$
$$sp(x := F, \alpha) = \neg x \wedge (\alpha[x \leftarrow T] \vee \alpha[x \leftarrow F])$$

3. **Instrukcja sekwencyjnego złożenia.** Jest ona postaci $S_1; S_2$, gdzie S_1 i S_2 są dowolnymi instrukcjami. Wykonanie jej polega na wykonaniu najpierw S_1, a następnie S_2.

   $wlp(S_1; S_2, \alpha) = wlp(S_1, wlp(S_2, \alpha))$;

   $sp(S_1; S_2, \alpha) = sp(S_2, sp(S_1, \alpha))$.

4. **Instrukcja alternatywy.** Jest ona postaci

   if $B_1 \to S_1$ [] $\cdots$ [] $B_n \to S_n$ **fi** (4)

   gdzie $B_1, \ldots, B_n$ są wyrażeniami boolowskimi, a $S_1, \ldots, S_n$ dowolnymi instrukcjami.

$B_1, \ldots, B_n$ nazywają się *dozorami*, a wyrażenia $B_i \to S_i$ *instrukcjami dozorowanymi*.

Instrukcję alternatywy (4) będziemy oznaczać $\mathbf{if} \Box_{i=1}^{n} B_i \to S_i \ \mathbf{fi}$ lub w skrócie jako IF.

Wykonanie instrukcji przebiega następująco: jeśli wszystkie dozory są fałszywe, wykonanie się kończy, w przeciwnym przypadku spośród tych instrukcji dozorowanych $B_i \to S_i$, dla których B_i jest prawdziwe zostaje *losowo* wybrana jedna i S_i zostaje wykonane.

$$wlp(IF, \alpha) = \bigwedge_{i=1}^{n}(B_i \Rightarrow wlp(S_i, \alpha)) \wedge ((\bigwedge_{i=1}^{n} \neg B_i) \Rightarrow \alpha)$$

$$sp(IF, \alpha) = (\bigvee_{i=1}^{n} sp(S_i, B_i \wedge \alpha)) \vee ((\bigwedge_{i=1}^{n} \neg B_i) \wedge \alpha).$$

Uwaga 1
W wersji oryginalnej języka Dijkstry przypadek, gdy wszystkie dozory są fałszywe sprowadza się do instrukcji *abort*, której nie włączyliśmy do naszego repertuaru. Podana tutaj wersja IF jest specjalnym przypadkiem IF Dijkstry, tj.: **if** $B_1 \to S_1 [] \cdots [] B_n \to S_n [] \neg B_1 \wedge \ldots \wedge \neg B_n \to skip$ **fi**.∎

Uwaga 2
Zauważmy, że jeśli więcej niż jeden dozór jest prawdziwy, wybór instrukcji dozorowanej jest niedeterministyczny.∎

Klasa obliczeń reprezentowana przez scenariusz $SC = \langle\alpha_0\rangle\ A_1; \langle\alpha_1\rangle\ A_2; \ldots; \langle\alpha_{n-1}\rangle\ A_n\ \langle\alpha_n\rangle$ jest to wyrażenie oznaczone jako

$$[A_1; \ldots; A_n](\alpha_0, \ldots, \alpha_n). \quad (5)$$

Stany scenariusza
Niech $ST^{\mathcal{SC}}(i)$, $0 \leq i \leq n$, będzie zbiorem wszystkich stanów spełniających następujący warunek:
dla każdego $\sigma \in ST^{\mathcal{SC}}(i)$, istnieje obliczenie $c \in [A_1; \ldots; A_n](\alpha_0, \ldots, \alpha_n)$ takie, że σ jest osiągany przez c po wykonaniu $A_1; \ldots; A_i$. Elementy $ST^{\mathcal{SC}}(i)$ będziemy nazywać *i-stanami* SC.

0-stany – stany początkowe
n-stany – stany końcowe.

Opis i-stanów

Niech WLP_i^{SC} i SP_i^{SC} będą formułami zdefiniowanymi indukcyjnie w sposób następujący:

$SP_0^{SC} = \alpha_0$;
$WLP_n^{SC} = \neg\alpha_n$
i dla $0 < k \leq n$
$SP_k^{SC} = \alpha_k \wedge sp(A_k, SP_{k-1}^{SC})$;
$WLP_{n-k}^{SC} = \neg\alpha_{n-k} \vee wlp(A_{n-k+1}, WLP_{n-k+1}^{SC})$.

Twierdzenie 2
Niech $SC = \langle\alpha_0\rangle\ A_1; \langle\alpha_1\rangle\ A_2; \ldots; \langle\alpha_{n-1}\rangle\ A_n\ \langle\alpha_n\rangle$. Zbiór stanów początkowych scenariusza SC opisuje formuła

$$D_0^{SC} = \neg WLP_0^{SC} \wedge \alpha_0. \quad (6)$$

Formułę D_0^{SC} będziemy nazywać *opisem stanu początkowego* scenariusza SC.

Twierdzenie 3
Niech $SC = \langle\alpha_0\rangle\ A_1; \langle\alpha_1\rangle\ A_2; \ldots; \langle\alpha_{n-1}\rangle\ A_n\ \langle\alpha_n\rangle$. Zbiór stanów końcowych scenariusza SC opisuje formuła

$$D_n^{SC} = SP_n^{SC}. \quad (7)$$

Twierdzenie 4
Zbiór $i-$stanów scenariusza SC opisuje formuła

$$D_i^{SC} = SP_i^{SC} \wedge \neg WLP_i^{SC} \quad (8)$$

■

Przykłady

1. **Fluenty:**
   a – indyk żyje;
   l – strzelba jest naładowana

2. **Działania:**
   $load$: $l := T$;

   - $wlp(load, \alpha) = \alpha[l \leftarrow T]$;
   - $sp(load, \alpha) = l \wedge (\alpha[l \leftarrow T] \vee \alpha[l \leftarrow F])$;

   $shoot$: **if** $l \rightarrow\ a := F; l := F$ [] $\neg l \rightarrow\ skip$ **fi**;

   - $wlp(shoot, \alpha) = (l \Rightarrow \alpha[a \leftarrow F, l \leftarrow F]) \wedge (\neg l \Rightarrow \alpha)$;
   - $sp(shoot, \alpha) = (\neg a \wedge \neg l \wedge \alpha[a \leftarrow T, l \leftarrow T]) \vee (\neg a \wedge \neg l \wedge \alpha[a \leftarrow F, l \leftarrow T]) \vee (\neg l \wedge \alpha)$;

   $wait$: $skip$;

   - $wlp(wait, \alpha) = sp(wait, \alpha) = \alpha$;

   $spin$: **if** $T \rightarrow\ l := T$ [] $T \rightarrow\ l := F$ **fi**;

   - $wlp(spin, \alpha) = \alpha[l \leftarrow T] \wedge \alpha[l \leftarrow F]$;
   - $sp(spin, \alpha) = \alpha[l \leftarrow T] \vee \alpha[l \leftarrow F]$;

Przykład 2 (Yale Shooting Scenario)

$YSS = \langle a \wedge \neg l \rangle load; wait; shoot \langle T \rangle.$

Oczekiwaną konkluzją jest, że po wykonaniu działań $load; wait; shoot$ indyk nie żyje i strzelba jest nienaładowana.
Obliczamy D_3^{YSS}.

$$\begin{aligned} D_3^{YSS} &= sp(load; wait; shoot, a \wedge \neg l) \\ &= \neg a \wedge \neg l. \end{aligned}$$

Ponieważ $D_3^{YSS} \vdash \neg a \wedge \neg l$, więc w stanie końcowym indyk jest nieżywy i strzelba rozładowana.
∎

Przykład 3 (Russian Turkey Scenario)
(RTS) jest przykładem problemu predykcji, w którym występują działania niedeterministycze.

$RTS = \langle a \wedge \neg l \rangle\ load; spin; shoot\ \langle T \rangle.$

Oczekiwaną konkluzją jest, że nie możemy stwierdzić czy w stanie końcowym indyk żyje czy nie.

Obliczamy D_3^{RTS}.

$$\begin{aligned} D_3^{RTS} &= sp(load; spin; shoot, a \wedge \neg l) \\ &= \neg l. \end{aligned}$$

Ponieważ zarówno $D_3^{RTS} \not\vdash a$ jak i $D_3^{RTS} \not\vdash \neg a$, nie można wywnioskować czy w stanie końcowym indyk żyje, czy nie. ∎