Weryfikacja modelowa *jest* analizą statyczną logiki modalnej

Marcin Sulikowski

MIMUW

15 grudnia 2010

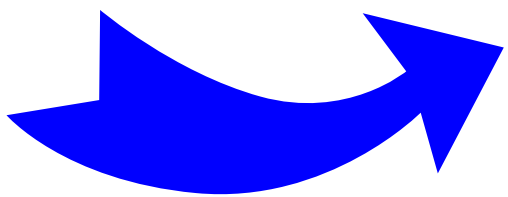
1. Wstęp
2. Weryfikacja systemów etykietowanych
3. Flow Logic
4. Weryfikacja modelowa $\subseteq$ analiza statyczna

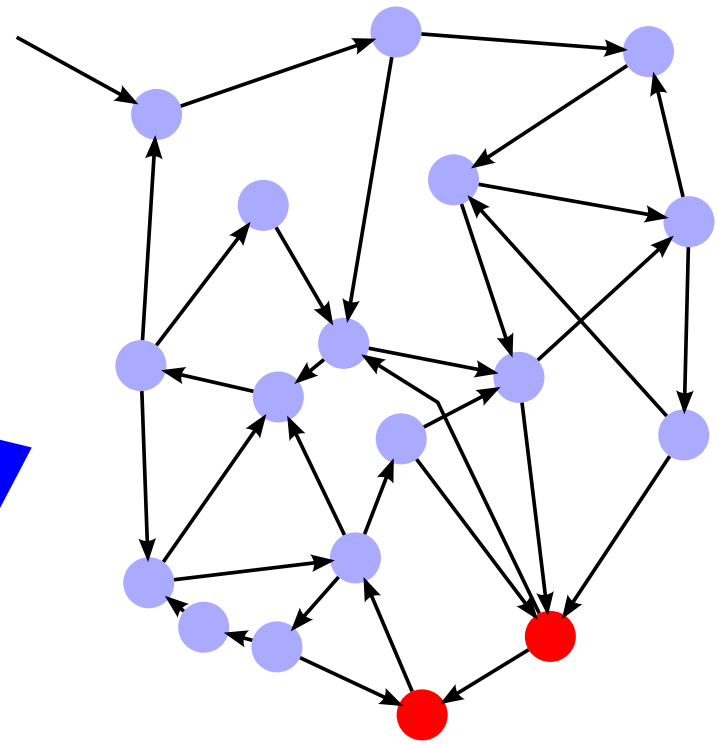
Co jest czym czego?

- Weryfikacja modelowa
- jest analizą statyczną
- logiki modalnej

Weryfikacja modelowa

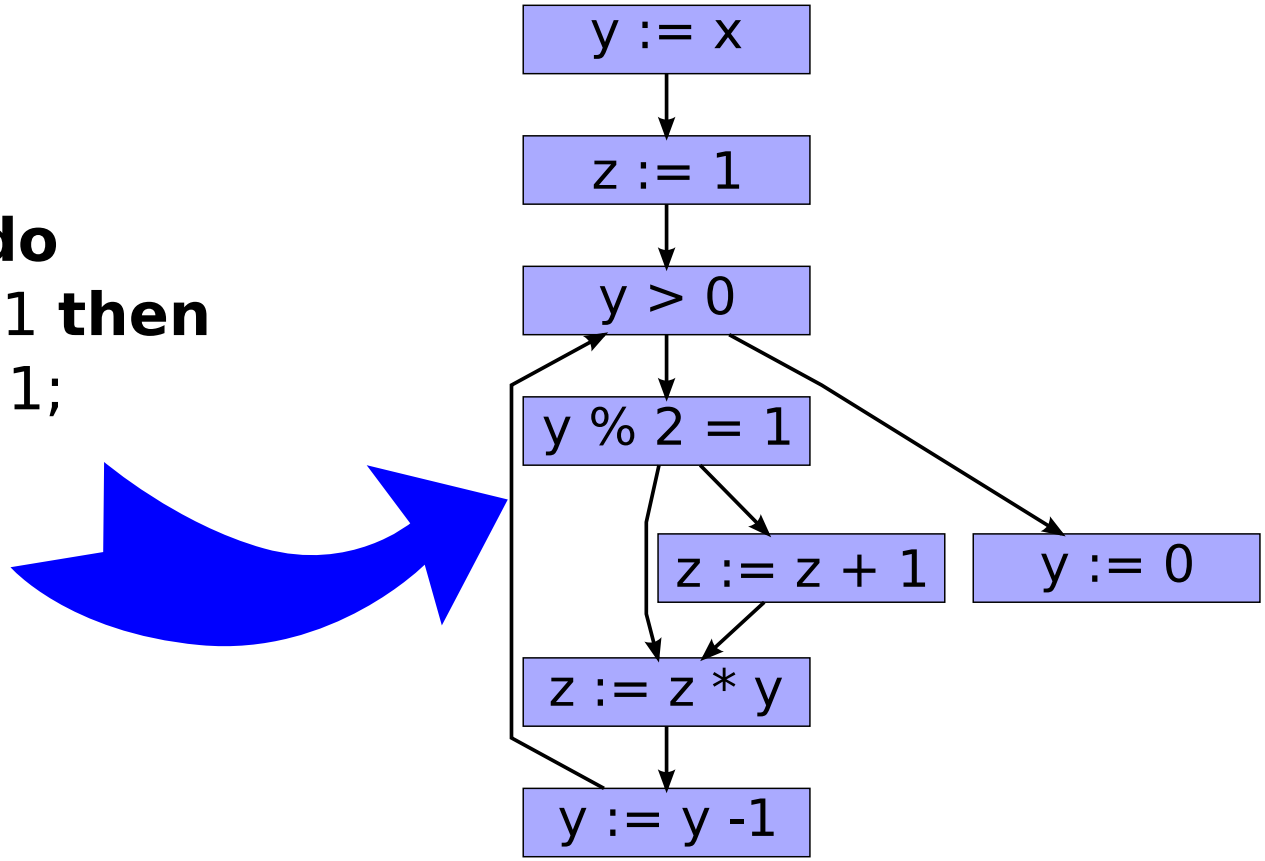
```
y := x;
z := 1;
while y > 0 do
    if y % 2 = 1 then
        z := z + 1;
    fi;
    z := z * y;
    y := y - 1;
od;
y := 0;
```





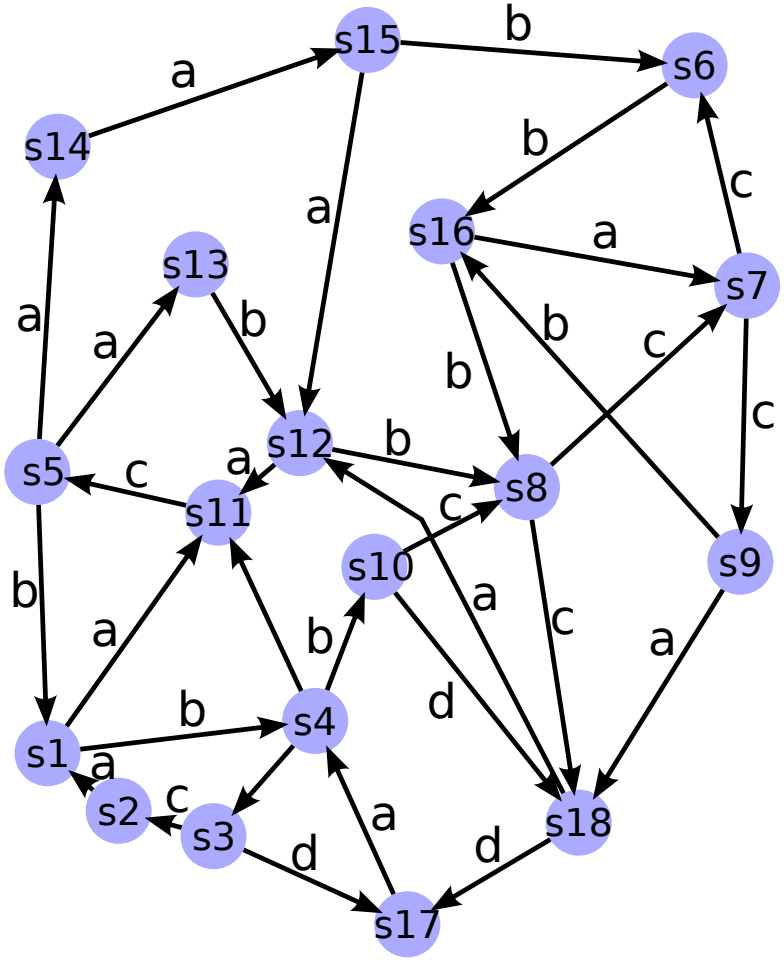
Analiza statyczna

```
y := x;
z := 1;
while y > 0 do
    if y % 2 = 1 then
        z := z + 1;
    fi;
    z := z * y;
    y := y - 1;
od;
y := 0;
```



- Analiza statyczna $\overset{?}{\subseteq}$ weryfikacja modelowa
- Analiza statyczna $\overset{?}{\supseteq}$ weryfikacja modelowa

System etykietowany



Zbiór stanów: $S = \{s1, \ldots, s18\}$

Zbiór akcji $\mathcal{A} = \{a, b, c, d\}$

Relacja przejścia $\rightarrow \subseteq S \times \mathcal{A} \times S$

ACTL

ACTL = Action Computation Tree Logic

$$\begin{aligned}\phi ::= \quad & \text{true} \mid \text{false} \mid bp \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \phi_1 \Rightarrow \phi_2 \\ & \mathbf{EX}_\Omega\phi \mid \mathbf{AX}_\Omega\phi \mid \mathbf{E}[\phi_1 \ {}_{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2] \mid \mathbf{A}[\phi_1 \ {}_{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2]\end{aligned}$$

$\Omega \subseteq \mathcal{A}$ opisują dopuszczalne akcje.

ACTL

Semantyka:

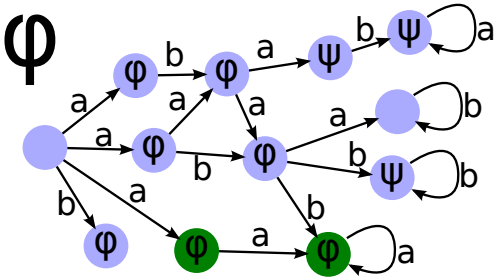
$$s_0 \models \mathbf{EX}_\Omega \phi \Leftrightarrow \quad \exists \left(s_i \rightarrow^{a_i} s_{i+1}\right)_{0 \leqslant i < n} : n > 0 \wedge a_0 \in \Omega \wedge s_1 \models \phi$$

$$s_0 \models \mathbf{AX}_\Omega \phi \Leftrightarrow \quad \forall \left(s_i \rightarrow^{a_i} s_{i+1}\right)_{0 \leqslant i < n} : n > 0 \wedge a_0 \in \Omega \wedge s_1 \models \phi$$

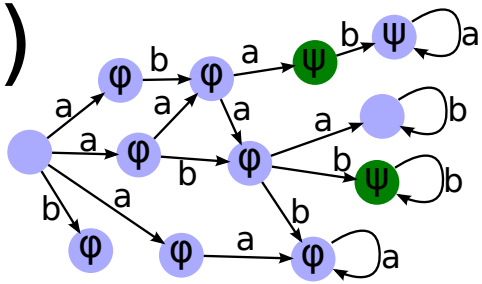
$$s_0 \models \mathbf{E}[\phi_1 \ {}_{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2] \Leftrightarrow \quad \exists \left(s_i \rightarrow^{a_i} s_{i+1}\right)_{0 \leqslant i < n} : \exists k < n : \left(\bigwedge_{0 \leqslant i < k} (a_i \in \Omega_1 \wedge s_{i+1} \models \phi_1) \wedge (a_k \in \Omega_2 \wedge s_{k+1} \models \phi_2)\right)$$

$$s_0 \models \mathbf{A}[\phi_1 \ {}_{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2] \Leftrightarrow \quad \forall \left(s_i \rightarrow^{a_i} s_{i+1}\right)_{0 \leqslant i < n} : \exists k < n : \left(\bigwedge_{0 \leqslant i < k} (a_i \in \Omega_1 \wedge s_{i+1} \models \phi_1) \wedge (a_k \in \Omega_2 \wedge s_{k+1} \models \phi_2)\right)$$

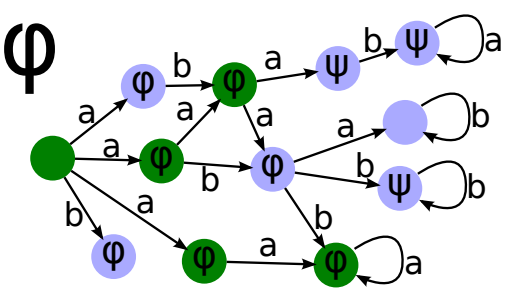
$\mathbf{AX}_A\,\varphi$



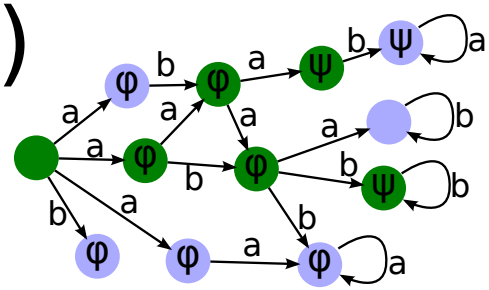
$\mathbf{A}(\varphi \ {}_A\mathbf{U}_B \ \psi)$



$\mathbf{EX}_A\,\varphi$



$\mathbf{E}(\varphi \ {}_A\mathbf{U}_B \ \psi)$



Flow Logic

- Podejście do **analizy statycznej** czerpiące m.in. z interpretacji abstrakcyjnej, analizy przepływu danych
- Za pomocą wyrażeń logicznych opisujemy **specyfikację poprawnego wyniku analizy**
- Jeśli zrobimy to dobrze, to da się efektywnie obliczyć wynik

Flow Logic na przykładzie

Analiza λ-wyrażeń

$$e ::= x \mid \lambda x.e \mid e_1\ e_2$$

$v \in \mathrm{Val}$ – zbiór wyrażeń w postaci: $v ::= x \mid \lambda x.e$

Relacja przejścia

- $(\lambda x.e)\ v \rightarrow^{(\lambda x.e, v)} e[v/x]$
- Jeśli $e_1 \rightarrow^{\beta} e_1'$, to: $e_1\ e_2 \rightarrow^{\beta} e_1'\ e_2$
- Jeśli $e_1 \rightarrow^{\beta} e_1'$, to: $e_2\ e_1 \rightarrow^{\beta} e_2\ e_1'$

Szczegóły techniczne

- Konwersja α
- Etykietowanie: $((\lambda x.e^{l'})^{l_1}\ v^{l_2})^{l} \rightarrow^{((\lambda x.e^{l'})^{l_1}, v^{l_2})} (e[v/x])^{l}$

Flow Logic na przykładzie

Cel analizy

$\mathcal{C} : \mathrm{Lab} \to \mathcal{P}(\mathrm{Val})$ – możliwe **wartości wyrażeń**

$\mathcal{R} : \mathrm{Var} \to \mathcal{P}(\mathrm{Val})$ – możliwe **wartości zmiennych**

Flow Logic na przykładzie

Cel analizy

$\mathcal{C} : \mathsf{Lab} \to \mathcal{P}(\mathsf{Val})$ – możliwe **wartości wyrażeń**
$\mathcal{R} : \mathsf{Var} \to \mathcal{P}(\mathsf{Val})$ – możliwe **wartości zmiennych**

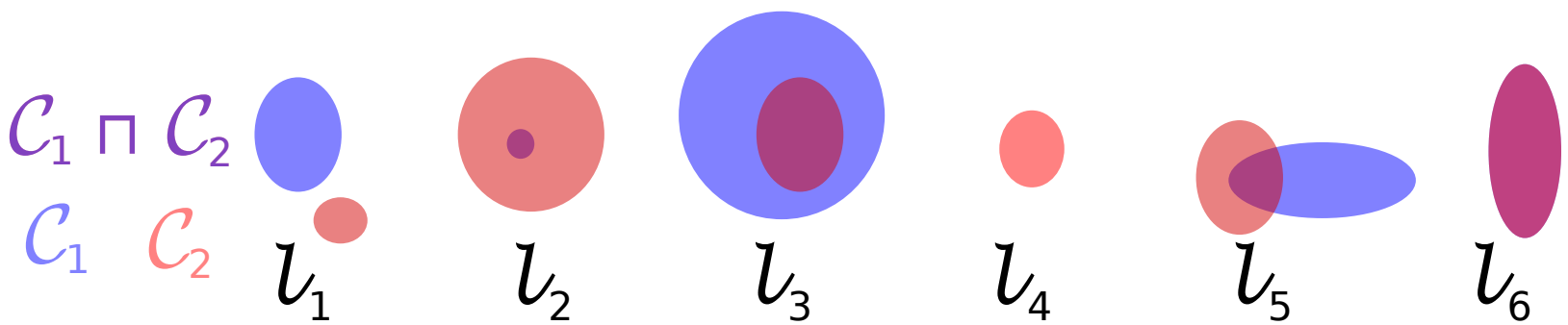
Specyfikacja

$$
\begin{aligned}
\mathcal{C}, \mathcal{R} \vdash x^l \Leftrightarrow \quad & \mathcal{R}(x) \subseteq \mathcal{C}(l) \\
\mathcal{C}, \mathcal{R} \vdash (\lambda x.e^{l'})^l \Leftrightarrow \quad & (\mathcal{C}, \mathcal{R} \vdash e^{l'}) \wedge (\lambda x.e^{l'}) \in \mathcal{C}(l) \\
\mathcal{C}, \mathcal{R} \vdash (e_1^{l_1}\ e_2^{l_2})^l \Leftrightarrow \quad & (\mathcal{C}, \mathcal{R} \vdash e_1^{l_1}) \wedge (\mathcal{C}, \mathcal{R} \vdash e_2^{l_2}) \wedge \\
& \forall (\lambda x.e_0^{l_0}) \in \mathcal{C}(l_1) : (\mathcal{C}(l_2) \subseteq \mathcal{R}(x) \wedge \mathcal{C}(l_0) \subseteq \mathcal{C}(l))
\end{aligned}
$$

Flow Logic na przykładzie

Obliczanie $(\mathcal{C}, \mathcal{R})$

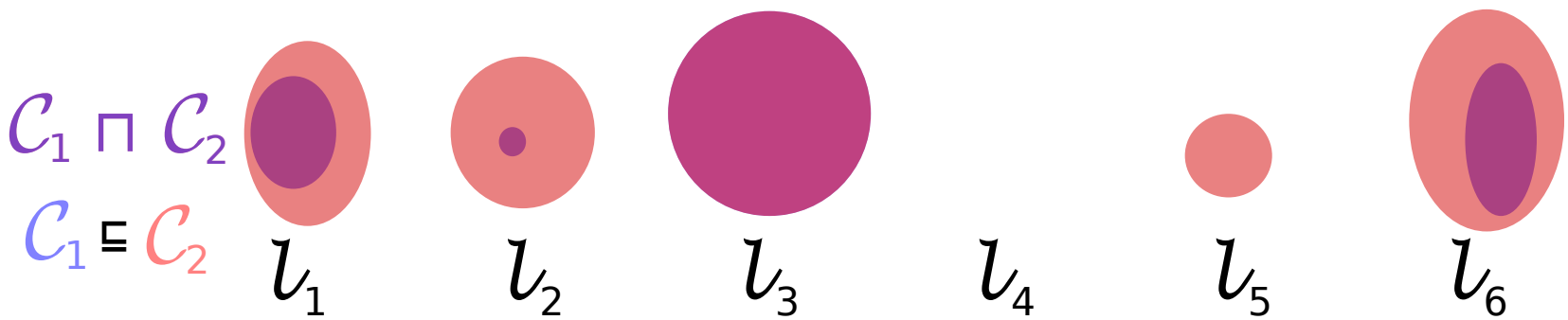
- $(\mathcal{C}, \mathcal{R})$ tworzą **kratę zupełną**
- Zbiór $\{(\mathcal{C}, \mathcal{R}) \mid \mathcal{C}, \mathcal{R} \vdash e^l\}$ jest zamknięty na branie infimum
- Istnieje więc **najlepsze (najmniejsze) rozwiązanie**
- Trzeba je tylko znaleźć...



Flow Logic na przykładzie

Obliczanie $(\mathcal{C}, \mathcal{R})$

- $(\mathcal{C}, \mathcal{R})$ tworzą **kratę zupełną**
- Zbiór $\left\{(\mathcal{C}, \mathcal{R}) \mid \mathcal{C}, \mathcal{R} \vdash e^{l}\right\}$ jest zamknięty na branie infimum
- Istnieje więc **najlepsze (najmniejsze) rozwiązanie**
- Trzeba je tylko znaleźć...



Flow Logic – ALFP

Alternation-free Least Fixed Point Logic

$$
\begin{array}{rl}
v ::= & c \mid x \mid f(v_1, \ldots, v_k) \\
pre ::= & R(v_1, \ldots, v_k) \mid \neg R(v_1, \ldots, v_k) \mid pre_1 \wedge pre_2 \mid pre_1 \vee pre_2 \mid \\
& \forall x : pre \mid \exists x : pre \\
cl ::= & R(v_1, \ldots, v_k) \mid true \mid cl_1 \wedge cl_2 \mid pre \Rightarrow cl \mid \forall x : cl
\end{array}
$$

Nie chcemy zmiennych wolnych. $\rho \models cl \iff (\rho, \lambda x.x) \models cl$

Flow Logic – ALFP

Alternation-free Least Fixed Point Logic

$$
\begin{array}{rl}
v ::= & c \mid x \mid f(v_1, \ldots, v_k) \\
pre ::= & R(v_1, \ldots, v_k) \mid \neg R(v_1, \ldots, v_k) \mid pre_1 \wedge pre_2 \mid pre_1 \vee pre_2 \mid \\
& \forall x : pre \mid \exists x : pre \\
cl ::= & R(v_1, \ldots, v_k) \mid true \mid cl_1 \wedge cl_2 \mid pre \Rightarrow cl \mid \forall x : cl
\end{array}
$$

Nie chcemy zmiennych wolnych. $\rho \models cl \iff (\rho, \lambda x.x) \models cl$

Ciekawe własności

- Powiemy, że dla pewnego warunku $R(v_1, \ldots, v_k)$ zachodzi
- Nie powiemy, że $R(v_1, \ldots, v_k)$ nie zachodzi

Flow Logic – ALFP

Alternation-free Least Fixed Point Logic

$$
\begin{array}{rl}
v ::= & c \mid x \mid f(v_1, \ldots, v_k) \\
pre ::= & R(v_1, \ldots, v_k) \mid \neg R(v_1, \ldots, v_k) \mid pre_1 \wedge pre_2 \mid pre_1 \vee pre_2 \mid \\
& \forall x : pre \mid \exists x : pre \\
cl ::= & R(v_1, \ldots, v_k) \mid true \mid cl_1 \wedge cl_2 \mid pre \Rightarrow cl \mid \forall x : cl
\end{array}
$$

Nie chcemy zmiennych wolnych. $\rho \models cl \iff (\rho, \lambda x.x) \models cl$

Ciekawe własności

- Powiemy, że dla pewnego warunku $R(v_1, \ldots, v_k)$ zachodzi
- Nie powiemy, że $R(v_1, \ldots, v_k)$ nie zachodzi
- Ale i tak trzeba uważać...

Flow Logic – ALFP

Twierdzenie o istnieniu rozwiązania

Zbiór $\{\rho : \rho \models cl\}$ jest zamknięty na branie infimum jeśli cl jest **warstwowa** i nie ma zmiennych wolnych. $\sqcap \{\rho : \rho \models cl\}$ jest najmniejszym modelem dla cl.

Jak znaleźć rozwiązanie?

`http://www2.imm.dtu.dk/cs_SuccinctSolver/`

ALFP – klauzula warstwowa

- Pozytywne użycie relacji R: $R(v_1, \ldots, v_k) \Rightarrow \ldots$
- Negatywne użycie relacji R: $\neg R(v_1, \ldots, v_k) \Rightarrow \ldots$
- Definicja relacji R: $\ldots \Rightarrow R(v_1, \ldots, v_k)$

ALFP – klauzula warstwowa

- Pozytywne użycie relacji R: $R(v_1, \ldots, v_k) \Rightarrow \ldots$
- Negatywne użycie relacji R: $\neg R(v_1, \ldots, v_k) \Rightarrow \ldots$
- Definicja relacji R: $\ldots \Rightarrow R(v_1, \ldots, v_k)$

Klauzula jest **warstwowa**, jeśli da się ją zapisać jako $cl_1 \wedge \ldots \wedge cl_r$ i dla każdej relacji istnieje taka liczba $0 \leqslant rank_R \leqslant r$, że:

- Definicje R występują w członach $cl_1, \ldots, cl_{rank_R}$
- Pozytywne użycia R występują w członach $cl_{rank_R} \ldots, n$
- Negatywne użycia R występują w członach $cl_{rank_R+1} \ldots, n$

$$
\begin{array}{cccc}
Pos\ R_1 & Neg\ R_1 & Neg\ R_1 & Neg\ R_1 \\
Def\ R_1 & Pos\ R_2 & Neg\ R_2 & Neg\ R_2 \\
Def\ R_2 \wedge & Def\ R_2 \wedge & Pos\ R_3 \wedge & Neg\ R_3 \\
Def\ R_3 & Def\ R_3 & Def\ R_3 & Neg\ R_3 \\
Def\ R_4 & Def\ R_4 & Def\ R_4 & Def\ R_4
\end{array}
$$

ALFP – przykład

Dla przypomnienia:

$$\begin{aligned}
\mathcal{C},\mathcal{R} \vdash x^l &\Leftrightarrow \quad \mathcal{R}(x) \subseteq \mathcal{C}(l) \\
\mathcal{C},\mathcal{R} \vdash (\lambda x.e^{l'})^l &\Leftrightarrow \quad (\mathcal{C},\mathcal{R} \vdash e^{l'}) \wedge (\lambda x.e^{l'}) \in \mathcal{C}(l) \\
\mathcal{C},\mathcal{R} \vdash (e_1^{l_1}\ e_2^{l_2})^l &\Leftrightarrow \quad (\mathcal{C},\mathcal{R} \vdash e_1^{l_1}) \wedge (\mathcal{C},\mathcal{R} \vdash e_2^{l_2}) \wedge \\
&\qquad \forall(\lambda x.e_0^{l_0}) \in \mathcal{C}(l_1) : (\mathcal{C}(l_2) \subseteq \mathcal{R}(x) \wedge \mathcal{C}(l_0) \subseteq \mathcal{C}(l))
\end{aligned}$$

To samo (?) w ALFP:

$$\begin{aligned}
\mathbf{C},\mathbf{R} \vdash x^l &\Leftrightarrow \quad \forall v : R_x(v) \Rightarrow C_l(v) \\
\mathbf{C},\mathbf{R} \vdash (\lambda x.e^{l'})^l &\Leftrightarrow \quad (\mathbf{C},\mathbf{R} \vdash e^{l'}) \wedge C_l(\lambda x.e^{l'}) \\
\mathbf{C},\mathbf{R} \vdash (e_1^{l_1}\ e_2^{l_2})^l &\Leftrightarrow \quad (\mathbf{C},\mathbf{R} \vdash e_1^{l_1}) \wedge (\mathbf{C},\mathbf{R} \vdash e_2^{l_2}) \wedge \\
&\qquad \Big[\forall(\lambda x.e_0^{l_0}) : \Big(\forall v : C_{l_1}(\lambda x.e_0^{l_0}) \wedge C_{l_2}(v) \Rightarrow R_x(v)\Big) \wedge \Big(\forall v' : C_{l_0}(v') \Rightarrow C_l(v')\Big)\Big]
\end{aligned}$$

Półmetek

Weryfikacja modelowa

- System z etykietowanymi akcjami
- Sprawdzamy, czy zachodzą formuły wyrażone w ACTL:

$$\begin{aligned}\phi ::= \quad & \text{true} \mid \text{false} \mid bp \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \phi_1 \Rightarrow \phi_2 \\ & \mathbf{EX}_{\Omega}\phi \mid \mathbf{AX}_{\Omega}\phi \mid \mathbf{E}[\phi_1 \ _{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2] \mid \mathbf{A}[\phi_1 \ _{\Omega_1}\mathbf{U}_{\Omega_2} \ \phi_2]\end{aligned}$$

Analiza statyczna

- Flow Logic, czyli specyfikacja w postaci: $\mathbf{R} \vdash e \Leftrightarrow \varphi$,
  gdzie e – element analizowanego języka, φ – formuła w ALFP
- Szukamy najmniejszego modelu spełniającego specyfikację

MC $\subseteq$ SA

Dane:

- S, $\mathcal{A}$, $\rightarrow \subseteq S \times \mathcal{A} \times S$
- $\phi \in$ ACTL

Szukamy:

- R_ϕ opisującej stany, w których spełnione jest ϕ
- $\phi \in$ ACTL

Szczegóły techniczne:

- Predykat $bp \rightsquigarrow$ relacja P_{bp}
- Podzbiory $\Omega \subseteq \mathcal{A} \rightsquigarrow$ relacje $\Omega \subseteq \mathcal{A}$
- Relacja $s \rightarrow^a s' \rightsquigarrow T(s, a, s')$
- Etykietujemy wszystkie podformuły ϕ

Flow Logic dla ACTL

$$\mathbf{R} \vdash \mathsf{true}^{l} \Leftrightarrow \quad \forall s : R_{\mathsf{true}^{l}}(s)$$

$$\mathbf{R} \vdash \mathsf{false}^{l} \Leftrightarrow \quad \mathsf{true}$$

$$\mathbf{R} \vdash bp^{l} \Leftrightarrow \quad \left(\forall s : P_{bp}(s) \Rightarrow R_{bp^{l}}(s)\right)$$

$$\mathbf{R} \vdash (\phi_1^{l_1} \wedge \phi_2^{l_2})^{l} \Leftrightarrow \quad \mathbf{R} \vdash \phi_1^{l_1} \wedge \mathbf{R} \vdash \phi_2^{l_2} \wedge \left(\forall s : R_{\phi_1^{l_1}}(s) \wedge R_{\phi_2^{l_2}}(s) \Rightarrow R_{(\phi_1^{l_1} \wedge \phi_2^{l_2})^{l}}(s)\right)$$

$$\mathbf{R} \vdash (\phi_1^{l_1} \vee \phi_2^{l_2})^{l} \Leftrightarrow \quad \mathbf{R} \vdash \phi_1^{l_1} \wedge \mathbf{R} \vdash \phi_2^{l_2} \wedge \left(\forall s : R_{\phi_1^{l_1}}(s) \vee R_{\phi_2^{l_2}}(s) \Rightarrow R_{(\phi_1^{l_1} \vee \phi_2^{l_2})^{l}}(s)\right)$$

$$\mathbf{R} \vdash (\neg \phi^{l'})^{l} \Leftrightarrow \quad \mathbf{R} \vdash \phi^{l'} \wedge \left(\forall s : \neg R_{\phi^{l'}}(s) \Rightarrow R_{\left(\neg \phi^{l'}\right)^{l}}(s)\right)$$

$$\mathbf{R} \vdash (\phi_1^{l_1} \Rightarrow \phi_2^{l_2})^{l} \Leftrightarrow \quad \mathbf{R} \vdash \phi_1^{l_1} \wedge \mathbf{R} \vdash \phi_2^{l_2} \wedge \left(\forall s : \neg R_{\phi_1^{l_1}}(s) \vee R_{\phi_2^{l_2}}(s) \Rightarrow R_{(\phi_1^{l_1} \Rightarrow \phi_2^{l_2})^{l}}(s)\right)$$

Flow Logic dla ACTL

$$\mathbf{R} \vdash \left(\mathbf{EX}_{\Omega}\phi^{l'}\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi^{l'} \wedge \left[\forall s : \left(\exists a : \exists s' : T(s,a,s') \wedge \Omega(a) \wedge R_{\phi^{l'}}(s')\right) \Rightarrow R_{\left(\mathbf{EX}_{\Omega}\phi^{l'}\right)^{l}}(s)\right]$$

$$\mathbf{R} \vdash \left(\mathbf{AX}_{\Omega}\phi^{l'}\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi^{l'} \wedge \left[\forall s : \left[\forall a : \forall s' : \neg T(s,a,s') \vee \left(\Omega(a) \wedge R_{\phi^{l'}}(s')\right)\right] \wedge \left[\exists a : \exists s' : T(s,a,s')\right] \Rightarrow R_{\left(\mathbf{AX}_{\Omega}\phi^{l'}\right)^{l}}(s)\right]$$

Flow Logic dla ACTL

$$\mathbf{R} \vdash \left(\mathbf{EX}_{\Omega}\phi^{l'}\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi^{l'} \wedge \left[\forall s : \left(\exists a : \exists s' : T(s,a,s') \wedge \Omega(a) \wedge R_{\phi^{l'}}(s')\right) \Rightarrow R_{\left(\mathbf{EX}_{\Omega}\phi^{l'}\right)^{l}}(s)\right]$$

$$\mathbf{R} \vdash \left(\mathbf{AX}_{\Omega}\phi^{l'}\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi^{l'} \wedge \left[\forall s : \left[\forall a : \forall s' : T(s,a,s') \Rightarrow \left(\Omega(a) \wedge R_{\phi^{l'}}(s')\right)\right] \wedge \left[\exists a : \exists s' : T(s,a,s')\right] \Rightarrow R_{\left(\mathbf{AX}_{\Omega}\phi^{l'}\right)^{l}}(s)\right]$$

Flow Logic dla ACTL

$$\mathbf{R} \vdash \left(\mathbf{E}\left[\phi_1^{l_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi_1^{l_1} \wedge \mathbf{R} \vdash \phi_2^{l_2} \wedge$$
$$\left[\forall s : \left(\exists a : \exists s' : T(s,a,s') \wedge \Omega_2(a) \wedge R_{\phi_2^{l_2}}(s')\right)\right.$$
$$\left.\Rightarrow R_{\left(\mathbf{E}\left[\phi_1^{l_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l}}(s)\right] \wedge$$
$$\left[\forall s : \left(\exists a : \exists s' : T(s,a,s') \wedge \Omega_1(a) \wedge R_{\phi_1^{l_1}}(s') \wedge\right.\right.$$
$$\left.\left. R_{\left(\mathbf{E}\left[\phi_1^{l_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l}}(s')\right) \Rightarrow R_{\left(\mathbf{E}\left[\phi_1^{l_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l}}(s)\right]$$

Flow Logic dla ACTL

$$\mathbf{R} \vdash \left(\mathbf{A}\left[\phi_1^{l_1}\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l} \Leftrightarrow \mathbf{R} \vdash \phi_1^{l_1} \wedge \mathbf{R} \vdash \phi_2^{l_2} \wedge$$

$$\forall s : \Bigg[(\exists a : \exists s' : T(s,a,s')) \wedge$$

$$\Big[\forall a : \forall s' : \neg T(s,a,s') \vee \left(\Omega_2(a) \wedge R_{\phi_2^{l_2}}(s')\right) \vee$$

$$\left(\Omega_1(a) \wedge R_{\phi_1^{l_1}}(s') \wedge R_{\left(\mathbf{A}\left[\phi_1^{l_1}\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l}}(s')\right)\Big]$$

$$\Rightarrow R_{\left(\mathbf{A}\left[\phi_1^{l_1}\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2^{l_2}\right]\right)^{l}}(s)\Bigg]$$

Rezultat

Najmniejszy model ρ spełnia dwa warunki:

- **Poprawność**: $s \models \phi^I \Rightarrow \rho\left(R_{\phi^I}\right)(s)$
- **Precyzja**: $\rho\left(R_{\phi^I}\right)(s) \Rightarrow s \models \phi^I$

Najmniejszy model znajdziemy w czasie $\mathcal{O}\left(\left(|T| + |S|\right)|\phi^I|\right)$

Bibliografia

F. Nielson, H. Riis Nielson
Model Checking Is Static Analysis of Modal Logic
FOSSACS, t. 6014 LNCS, ss. 191-205. Springer, 2010.

H. Riis Nielson, F. Nielson
Flow Logic: a multi paradigmatic approach to static analysis
The Essence of Computation, t. 2566 LNCS, strony 69-83. Springer, 2002.