Penser à lancer la démo...

Réduction de conditions de vérification

Une approche basée sur les graphes

Nicolas Stouls

Post-doctorant INRIA Saclay – Île-de-France, équipe-projet ProVal

Travaux réalisés en partenariat avec JF. Couchot et A. Giorgetti du LIFC

16 juin 2008

Introduction

Preuve de programmes

- Vérification d'un programme par rapport à une spécification
- Approche classique : *Vérification déductive*
  - Programme dont on a le code
  - Spécification sous forme d'annotations
    - Pré/post-conditions
    - Invariants
    - Assertions
  - L'analyse du code génère des « *conditions de vérification* »
    - CV : *aussi appelée obligations de preuve*

      Formule logique qui, si elle est vérifiée, permet de garantir le respect d'une propriété par le code.

Introduction

Exemple d'autres approches de preuve de programmes

- Analyse déductive
  - Pas de spécification nécessaire
  - Uniquement basé sur l'analyse du code
  - Exemple : *vérification de l'absence de buffer overflow*
- Vérification externe
  - Pas de code : seulement les API et la spécification
  - Vérification de la cohérence de différentes spécifications
  - Exemple : *vérification de compatibilité de modules*

Approche existante dans d'autres langages

Java : ESC/Java, basé sur le langage d'annotations JML

C# : Boogie, basé sur le langage d'annotations Spec#

C : Why/Caduceus, basé sur le langage d'annotations CML

Environnement : Plate-forme Why/Caduceus

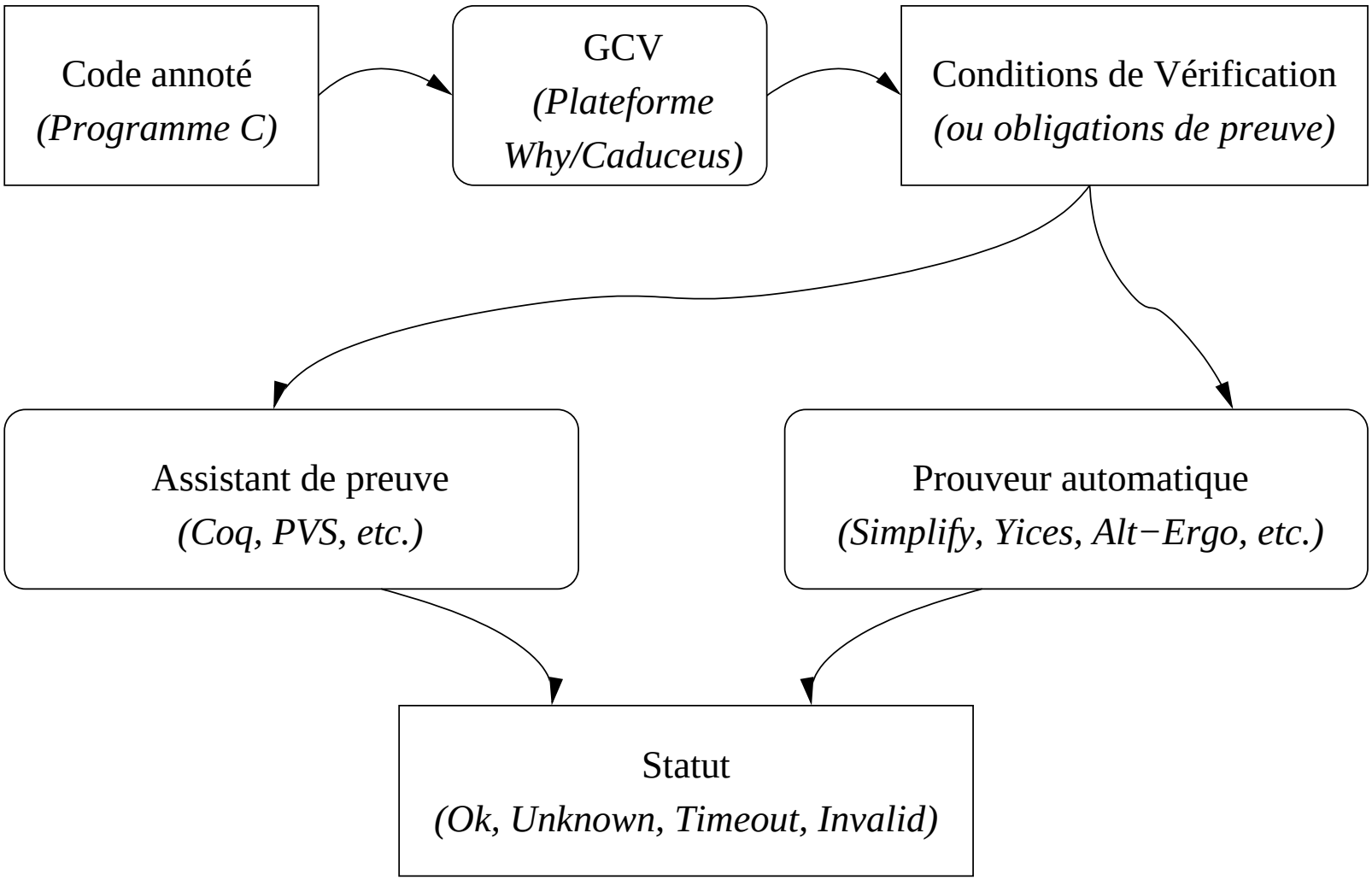
Problématiques liées au langage C

- Langage faiblement typé
- Norme ANSI non complète
- Arithmétique de pointeurs
- Aliasing possible
- *etc.*

Solutions déjà proposées par Why/Caduceus

- Induction de types
- Quelques restrictions sur le langage
- Modèle mémoire de *Burstall-Bornat*
  - Ramène la détection d'aliasing à du typage

Description générale du processus



Contexte

Cadre de ce travail

- Projet RNTL CAT (C Analyzis Toolbox)
  - Outils pour la validation par la preuve de programmes C
  - Plate-forme Caduceus/Why
- Projet System@tic PFC (Plates-formes de confiance)
  - Différentes contributions dont :
    - Utilisation de la plate-forme Caduceus/Why
    - Amélioration de l'automatisation des preuves

Notre objectif : maximiser l'automatisation des preuves

Différentes approches

- Optimisation de l'axiomatisation du langage
  - Par exemple le modèle mémoire
- Tactiques de simplification des CVs
  - Limitation de l'explosion combinatoire
  - Méthodes ad-hoc de réécriture des CVs *(liées aux prouveurs visés)*
  - Analyse et simplification propositionnelle des hypothèses

Notre approche : sélection d'hypothèses

- Principe : *Retirer les hypothèses non pertinentes*
- Définition intuitive de la pertinence :
  Une hypothèse h est pertinente si la condition de vérification est valide, mais qu'elle ne l'est plus si l'on retire h.
  $\Longrightarrow$ *Problème indécidable*
- Idée d'approximation :
  caractériser la pertinence d'une hypothèse par rapport à sa « distance » de la conclusion
- Approche par construction de graphes de dépendances

Exemple fil rouge

```
void swap(int Tab[], int a,int taille){
  Tab[a]=Tab[a]+Tab[a+1];
  Tab[a+1]=Tab[a]-Tab[a+1];
  Tab[a]=Tab[a]-Tab[a+1];
}
```

Exemple fil rouge

```
/*@ requires // Pré-condition
  @   0≤a<taille-1   ∧   \valid_range(Tab,0,taille-1)
  @ assigns // Déclaration des effets de bord
  @   Tab[a],Tab[a+1]
  @ ensures // Post-condition
  @   Tab[a] == \old(Tab[a+1])   ∧   Tab[a+1] == \old(Tab[a])
*/

void swap(int Tab[], int a,int taille){
  Tab[a]=Tab[a]+Tab[a+1];
  Tab[a+1]=Tab[a]-Tab[a+1];
  Tab[a]=Tab[a]-Tab[a+1];
}
```

Exemple fil rouge

```
/*@ requires // Pré-condition
  @   0≤a<taille-1   ∧   \valid_range(Tab,0,taille-1)
  @ assigns // Déclaration des effets de bord
  @   Tab[a],Tab[a+1]
  @ ensures // Post-condition
  @   Tab[a] == \old(Tab[a+1])   ∧   Tab[a+1] == \old(Tab[a])
*/

void swap(int Tab[], int a,int taille){
  Tab[a]=Tab[a]+Tab[a+1];
  Tab[a+1]=Tab[a]-Tab[a+1];
  Tab[a]=Tab[a]-Tab[a+1];
}
```

Certains prouveurs n'arrivent pas à établir la préservation de la post-condition de cette fonction.

Illustration du modèle mémoire de Caduceus

- Chaque pointeur pointe dans une mémoire
- Différentes mémoires caractérisent différents moments

void swap(**int** *Tab*[], **int** a,**int** taille){
// Tab est initialement dans la mémoire m
Tab[a]=Tab[a]+Tab[a+1] ;
// Tab modifié est dans la mémoire m_0
// Propriété : m et m_0 *ne diffèrent que part* $Tab + a$
Tab[a+1]=Tab[a]-Tab[a+1] ;
// Tab modifié est dans la mémoire m_1
// Propriété : m_0 *et* m_1 *ne diffèrent que part* $Tab + a + 1$
Tab[a]=Tab[a]-Tab[a+1] ;
// Tab modifié est dans la mémoire m_2
// Propriété : m_1 *et* m_2 *ne diffèrent que part* $Tab + a$
}

Problème Établir que m et m_2 ne diffèrent que par $Tab + a$ et $Tab + a + 1$

Illustration du modèle mémoire de Caduceus

- Chaque pointeur pointe dans une mémoire
- Différentes mémoires caractérisent différents moments

```
void swap(int Tab[], int a,int taille){
// Tab est initialement dans la mémoire m
  Tab[a]=Tab[a]+Tab[a+1];
// Tab modifié est dans la mémoire m_0
// Propriété : m et m_0 ne diffèrent que part Tab + a
  Tab[a+1]=Tab[a]-Tab[a+1];
// Tab modifié est dans la mémoire m_1
// Propriété : m_0 et m_1 ne diffèrent que part Tab + a + 1
  Tab[a]=Tab[a]-Tab[a+1];
// Tab modifié est dans la mémoire m_2
// Propriété : m_1 et m_2 ne diffèrent que part Tab + a
}
```

Problème Établir que m et m_2 ne diffèrent que par $Tab + a$ et $Tab + a + 1$

Illustration du modèle mémoire de Caduceus

- Chaque pointeur pointe dans une mémoire
- Différentes mémoires caractérisent différents moments

```
void swap(int Tab[], int a,int taille){
// Tab est initialement dans la mémoire m
  Tab[a]=Tab[a]+Tab[a+1];
// Tab modifié est dans la mémoire m_0
// Propriété : m et m_0 ne diffèrent que part Tab + a
  Tab[a+1]=Tab[a]-Tab[a+1];
// Tab modifié est dans la mémoire m_1
// Propriété : m_0 et m_1 ne diffèrent que part Tab + a + 1
  Tab[a]=Tab[a]-Tab[a+1];
// Tab modifié est dans la mémoire m_2
// Propriété : m_1 et m_2 ne diffèrent que part Tab + a
}
```

Problème Établir que m et m_2 ne diffèrent que par $Tab + a$ et $Tab + a + 1$

Illustration du modèle mémoire de Caduceus

- Chaque pointeur pointe dans une mémoire
- Différentes mémoires caractérisent différents moments

void swap(**int** *Tab*[], **int** a,**int** taille){
// *Tab est initialement dans la mémoire m*

Tab[a]=Tab[a]+Tab[a+1] ;
// *Tab modifié est dans la mémoire* m_0
// *Propriété : m et* m_0 *ne diffèrent que part* $Tab + a$

Tab[a+1]=Tab[a]-Tab[a+1] ;
// *Tab modifié est dans la mémoire* m_1
// *Propriété :* m_0 *et* m_1 *ne diffèrent que part* $Tab + a + 1$

Tab[a]=Tab[a]-Tab[a+1] ;
// *Tab modifié est dans la mémoire* m_2
// *Propriété :* m_1 *et* m_2 *ne diffèrent que part* $Tab + a$
}

Problème Établir que m et m_2 ne diffèrent que par $Tab + a$ et $Tab + a + 1$

Détail d'une CV (Suite de l'exemple fil rouge)

$$
\begin{array}{l}
\quad h_1 \\
\wedge\, h_2 \\
\wedge\, \ldots \\
\wedge\, \ldots \\
\wedge\, h_n \\
\wedge\, h_{n+1} \\
\wedge\, h_{n+2} \\
\wedge\, \ldots \\
\wedge\, \ldots \\
\wedge\, h_m \\
\Longrightarrow \\
\quad \mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\}) \;\; \textit{// But}
\end{array}
$$

Détail d'une CV (Suite de l'exemple fil rouge)

$$
\left.\begin{array}{l} h_1 \\ \wedge\ h_2 \\ \wedge\ \ldots \\ \wedge\ \ldots \\ \wedge\ h_n \end{array}\right\} \textit{Axiomatisation}
$$

$$
\left.\begin{array}{l} \wedge\ h_{n+1} \\ \wedge\ h_{n+2} \\ \wedge\ \ldots \\ \wedge\ \ldots \\ \wedge\ h_m \end{array}\right\} \textit{Contexte}
$$

$$
\Longrightarrow
$$

$$
\mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\}) \quad \textit{// But}
$$

Détail d'une CV (Suite de l'exemple fil rouge)

$$
\left.\begin{array}{l} h_1 \\ \wedge\, h_2 \\ \wedge\, \ldots \\ \wedge\, \ldots \\ \wedge\, h_n \end{array}\right\} \textit{Axiomatisation}
$$

$$
\left.\begin{array}{l} \wedge\, h_{n+1} \\ \wedge\, m_0 = \mathit{upd}\left(\begin{array}{l} m \;,\quad \mathit{Tab}+a \;, \\ \mathit{acc}(m, \mathit{Tab}+a) \;+\; \mathit{acc}(m, \mathit{Tab}+a+1) \end{array}\right) \\ \wedge\, \ldots \\ \wedge\, h_m \end{array}\right\} \textit{Contexte}
$$

$$
\Longrightarrow
$$

$$
\mathbf{diff}(m, m_2, \{\mathit{Tab}+a+1, \mathit{Tab}+a\}) \quad \textit{// But}
$$

Détail d'une CV (Suite de l'exemple fil rouge)

$$
\begin{array}{l}
\quad h_1 \\
\wedge\ \mathbf{diff}(m_1, m_2, lp) \Leftrightarrow \forall p \cdot \left(\begin{array}{c} \backslash\mathbf{valid}(p)\ \wedge\ \neg\mathbf{mem}(p, lp) \\ \Rightarrow acc(m_1, p) = acc(m_2, p) \end{array} \right) \\
\wedge \ldots \\
\wedge\ h_n
\end{array}
\Bigg\} \textit{Axiomatisation}
$$

$$
\begin{array}{l}
\wedge\ h_{n+1} \\
\wedge\ m_0 = upd \left(\begin{array}{l} m\ ,\quad Tab{+}a\ , \\ acc(m, Tab{+}a)\ +\ acc(m, Tab{+}a{+}1) \end{array} \right) \\
\wedge \ldots \\
\wedge\ h_m
\end{array}
\Bigg\} \textit{Contexte}
$$

$$\Longrightarrow$$

$\quad \mathbf{diff}(m, m_2, \{Tab{+}a{+}1, Tab{+}a\})$ *// But*

Graphes de dépendance

Idée

Caractériser la distance entre une hypothèse et le but par :

- la distance entre leurs prédicats (**diff**, $\backslash$**valid**, **mem**, *etc.*)
- la distance entre leurs termes (*Tab*, m_i, *etc.*)

Noeuds

Termes ou prédicats (suivant le type de graphe)

Liaisons des Noeuds

ou $\left.\begin{array}{l}\text{2 prédicats}\\ \text{2 termes}\end{array}\right\}$ sont liées s'ils existent dans une même hypothèse

Construction d'un graphe de dépendance des termes

- Construit à partir des hypothèses du contexte
- Méthode proposée :
  1. Pour chaque hypothèse du contexte

$$m_0 = upd\left(\ m,\ \ Tab{+}a,\ \ acc(m, Tab{+}a)\ +\ acc(m, Tab{+}a{+}1)\ \right)$$

  2. On aplatit ses termes
     (chaque fonction « génère » une variable fraîche)

$$\begin{aligned} & m_0 = upd_0 \\ & \wedge\ upd_0 = upd\left(\ m,\ \ shift_0,\ \ shift_1\ \right) \\ & \wedge\ shift_0 = Tab{+}a \\ & \wedge\ shift_1 = acc_0\ +\ acc_1 \\ & \wedge\ acc_0 = acc(m, shift_0) \\ & \wedge\ acc_1 = acc(m, shift_2) \\ & \wedge\ shift_2 = Tab{+}a{+}1 \end{aligned}$$

  3. Chaque littéral est caractérisé par ses termes non fonctionnels

Construction d'un graphe de dépendance des termes

- Construit à partir des hypothèses du contexte
- Méthode proposée :
  1. Pour chaque hypothèse du contexte

$$m_0 = upd\left(\ m,\ \ Tab{+}a,\ \ acc(m, Tab{+}a)\ +\ acc(m, Tab{+}a{+}1)\ \right)$$

  2. On aplatit ses termes
     (chaque fonction « génère » une variable fraîche)

$$\begin{aligned} & m_0 = upd_0 \\ & \wedge\ upd_0 = upd\left(\ m,\ \ shift_0,\ \ shift_1\ \right) \\ & \wedge\ shift_0 = Tab{+}a \\ & \wedge\ shift_1 = acc_0\ +\ acc_1 \\ & \wedge\ acc_0 = acc(m, shift_0) \\ & \wedge\ acc_1 = acc(m, shift_2) \\ & \wedge\ shift_2 = Tab{+}a{+}1 \end{aligned}$$

  3. Chaque littéral est caractérisé par ses termes non fonctionnels

Construction d'un graphe de dépendance des termes

- Construit à partir des hypothèses du contexte
- Méthode proposée :
  1. Pour chaque hypothèse du contexte

$$m_0 = upd\left(\ m,\ \ Tab+a,\ \ acc(m, Tab+a)\ +\ acc(m, Tab+a+1)\ \right)$$

  2. On aplatit ses termes
     (chaque fonction « génère » une variable fraîche)

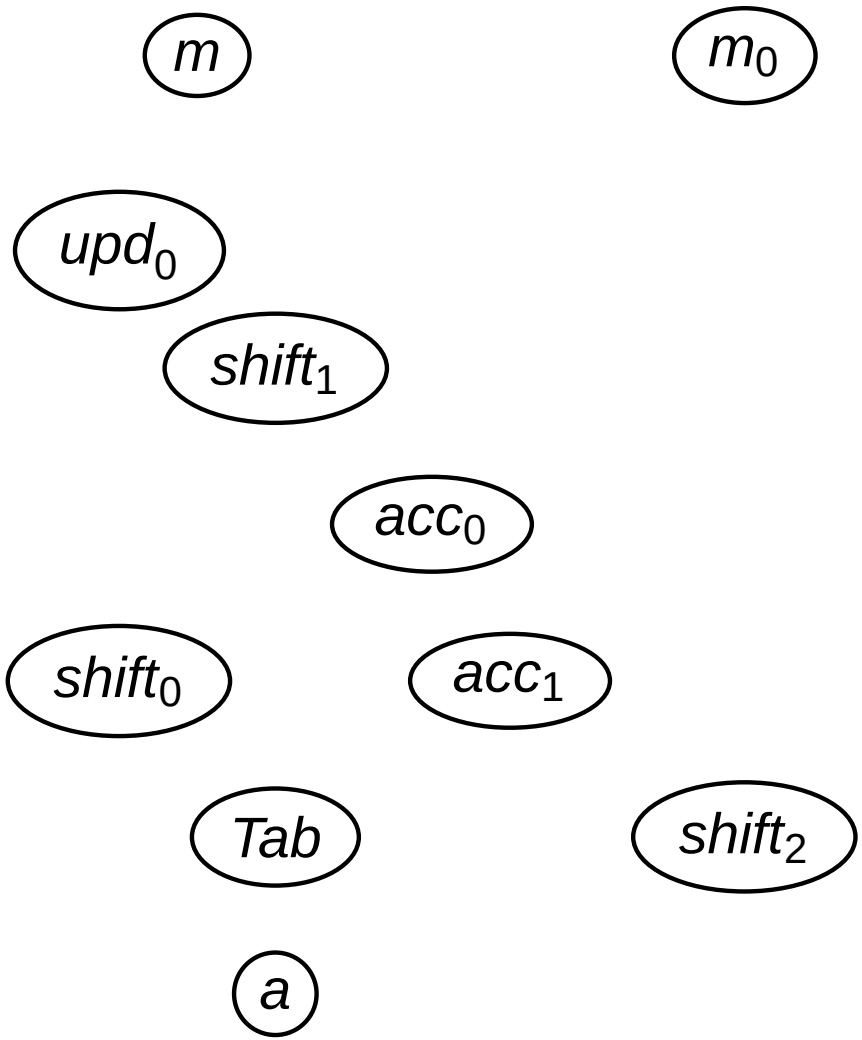
$$\begin{array}{ll}
m_0 = upd_0 & \{m_0, upd_0\} \\
\wedge\ upd_0 = upd\left(\ m,\ \ shift_0,\ \ shift_1\ \right) & \{upd_0, m, shift_0, shift_1\} \\
\wedge\ shift_0 = Tab+a & \{shift_0, Tab, a\} \\
\wedge\ shift_1 = acc_0\ +\ acc_1 & \{shift_1, acc_0, acc_1\} \\
\wedge\ acc_0 = acc(m, shift_0) & \{acc_0, m, shift_0\} \\
\wedge\ acc_1 = acc(m, shift_2) & \{acc_1, m, shift_2\} \\
\wedge\ shift_2 = Tab+a+1 & \{shift_2, Tab, a\}
\end{array}$$

  3. Chaque littéral est caractérisé par ses termes non fonctionnels

Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

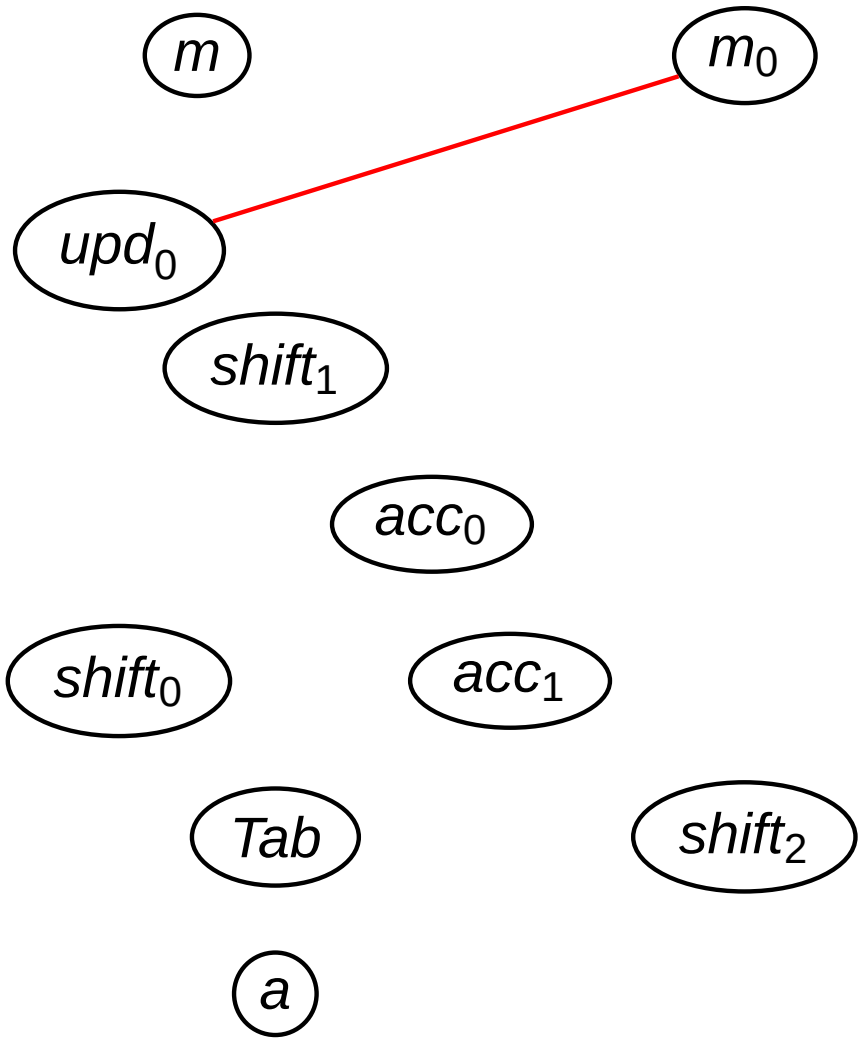
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

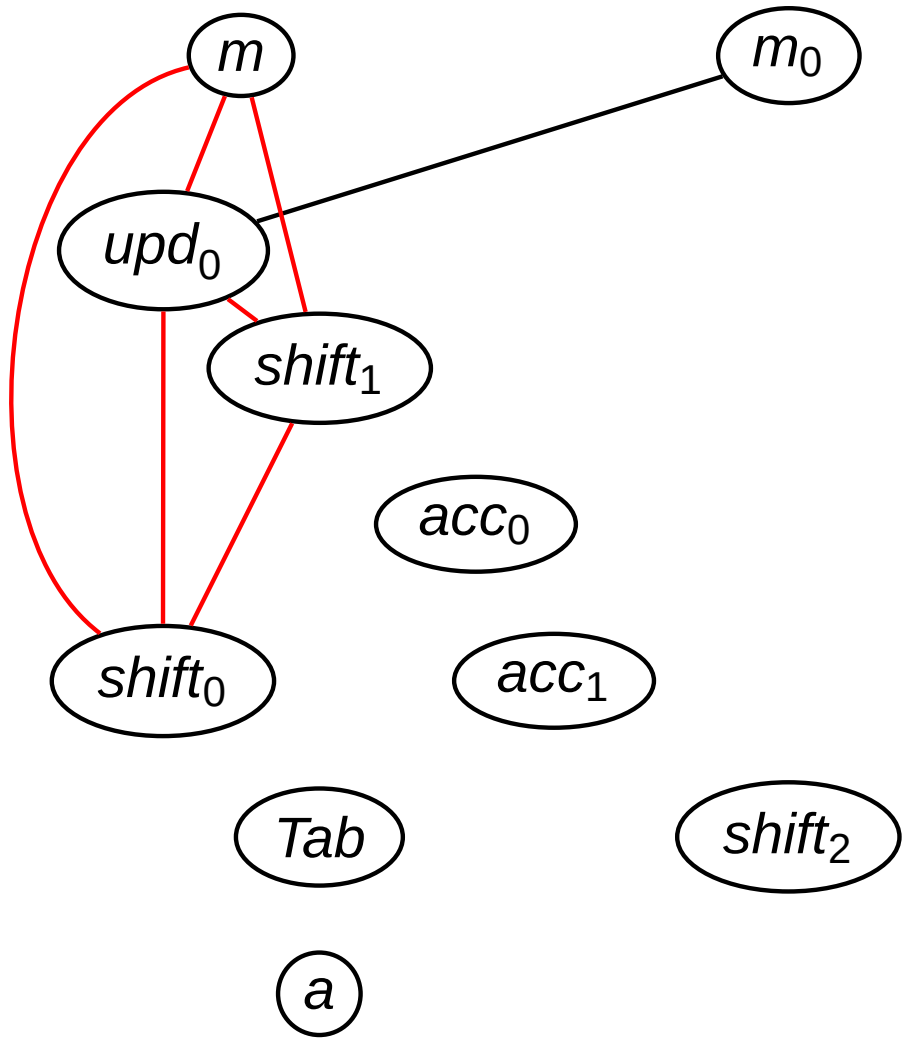
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

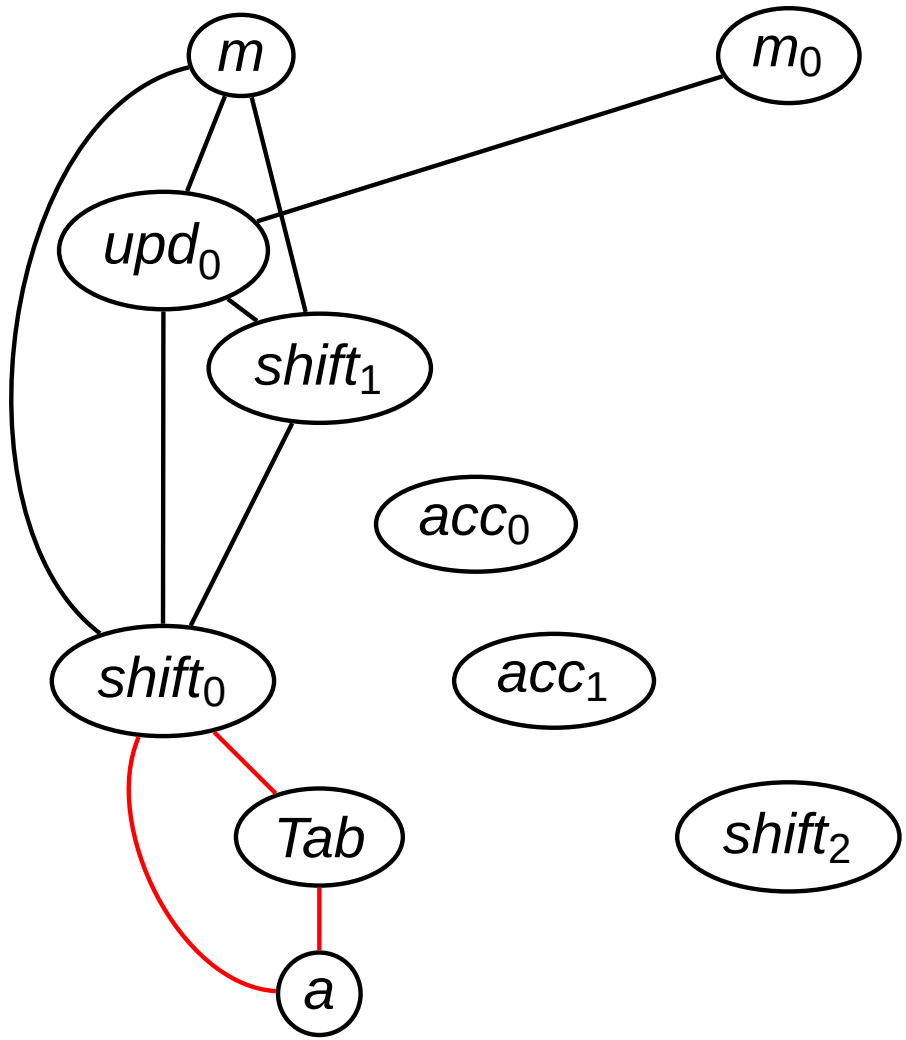
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

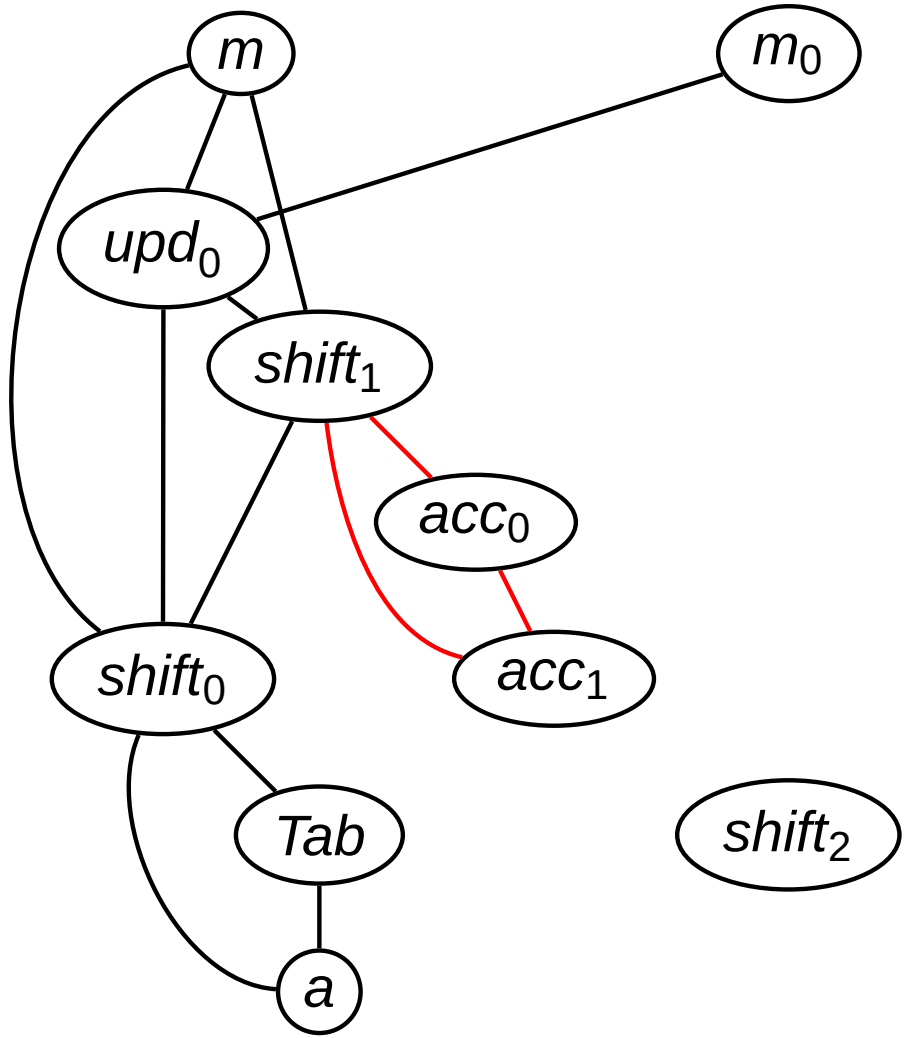
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

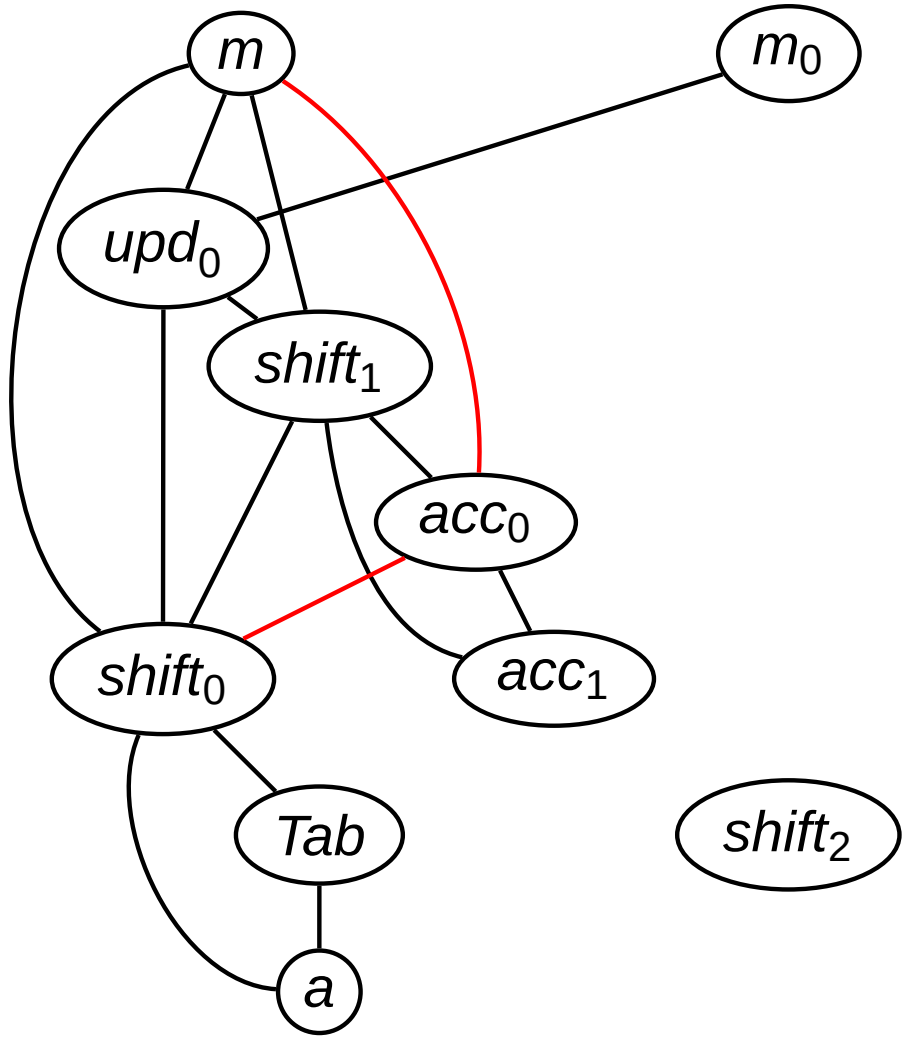
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

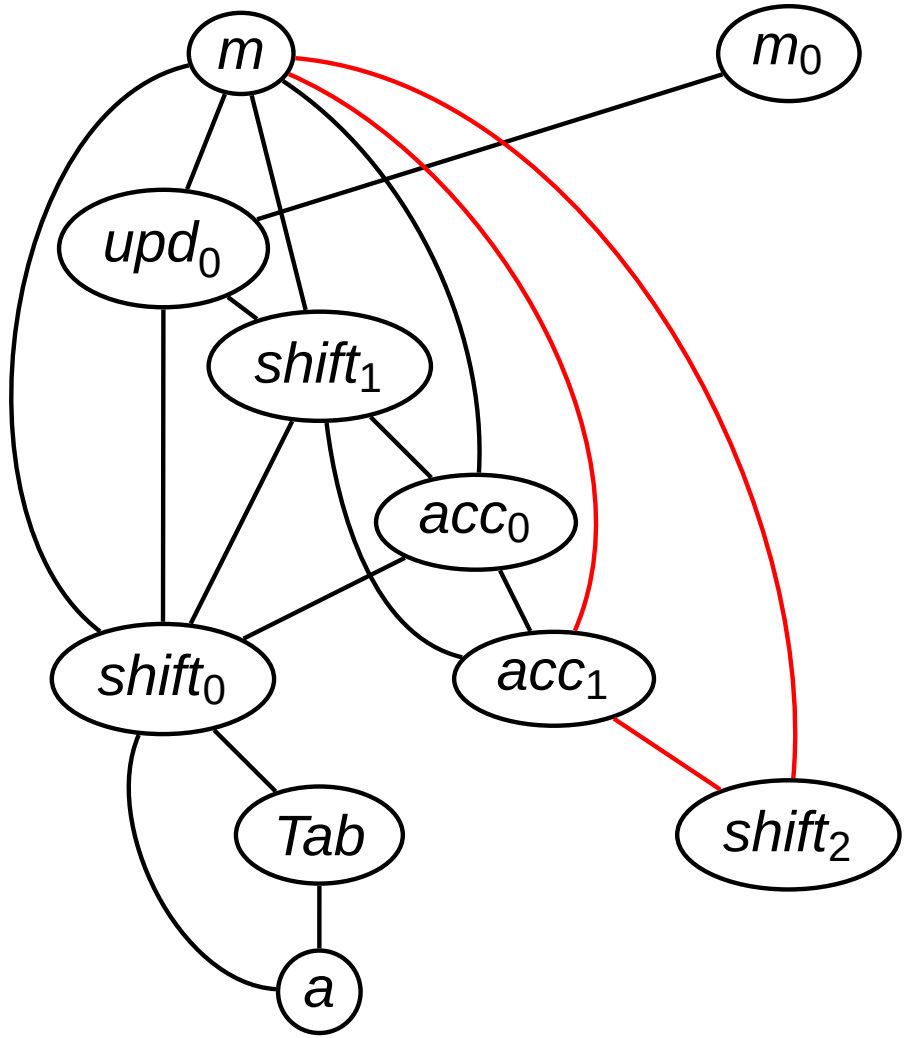
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

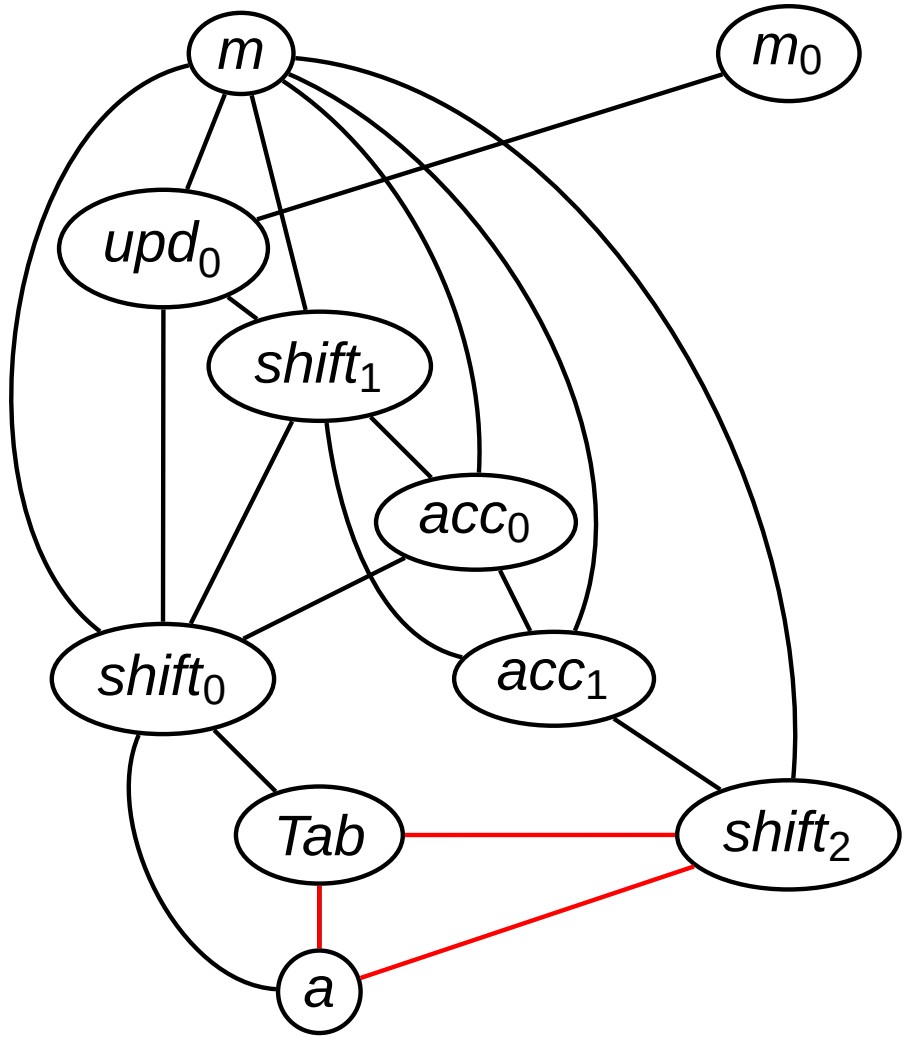
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

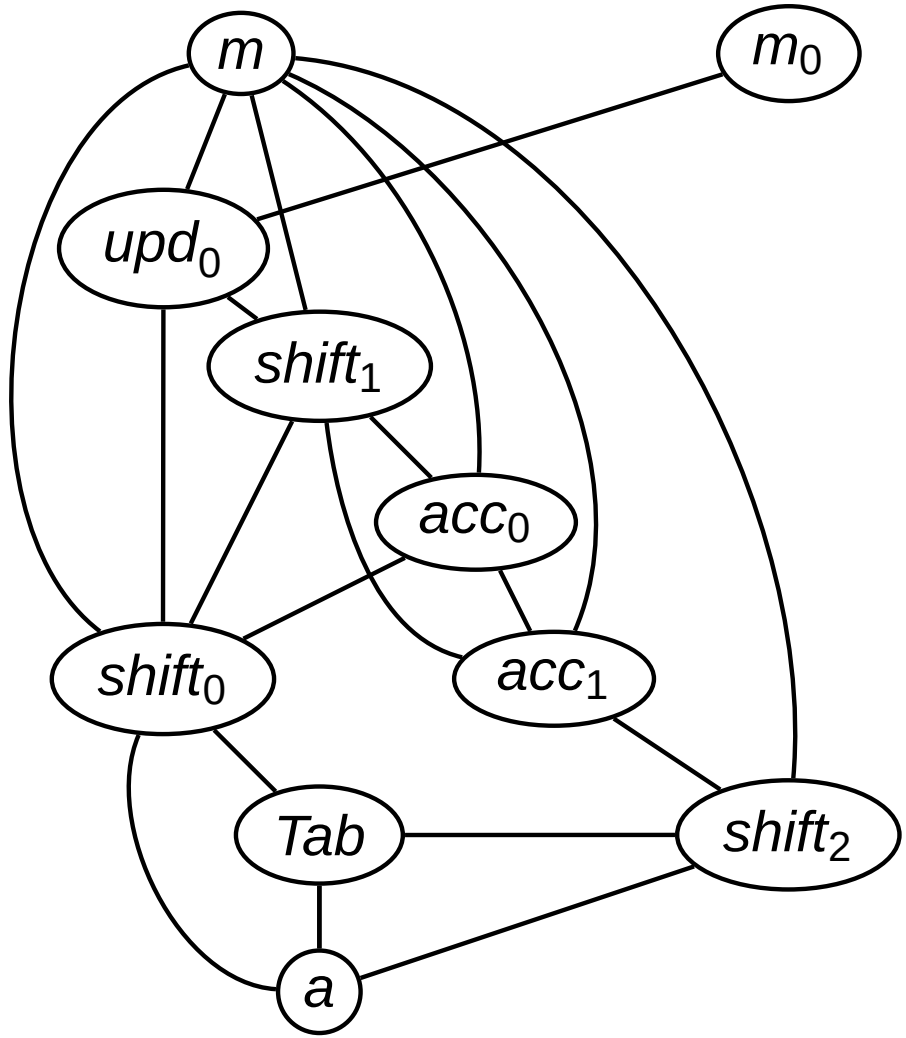
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Graphe de dépendance des termes

4. Chaque terme non fonctionnel devient un noeud
5. Chaque ensemble de termes est fortement connecté

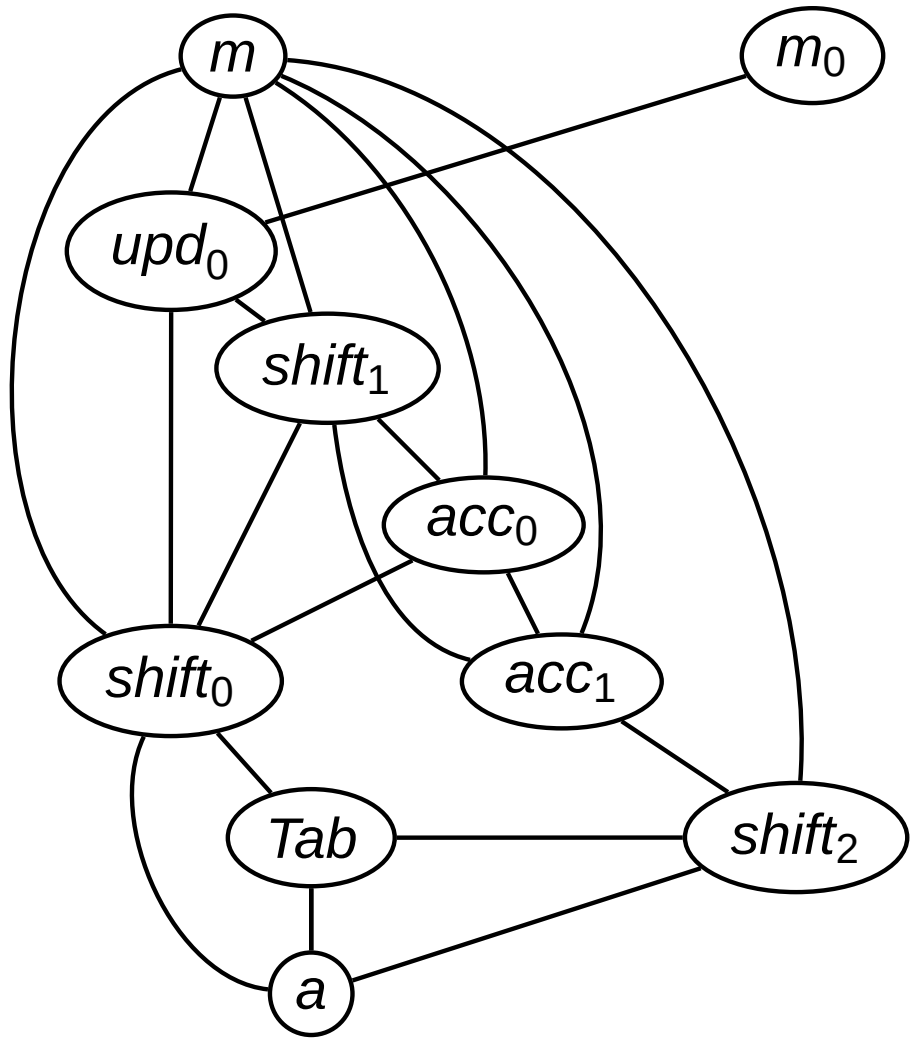
- $\{m_0, upd_0\}$
- $\{upd_0, m, shift_0, shift_1\}$
- $\{shift_0, Tab, a\}$
- $\{shift_1, acc_0, acc_1\}$
- $\{acc_0, m, shift_0\}$
- $\{acc_1, m, shift_2\}$
- $\{shift_2, Tab, a\}$



Degré de pertinence d'un terme

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$

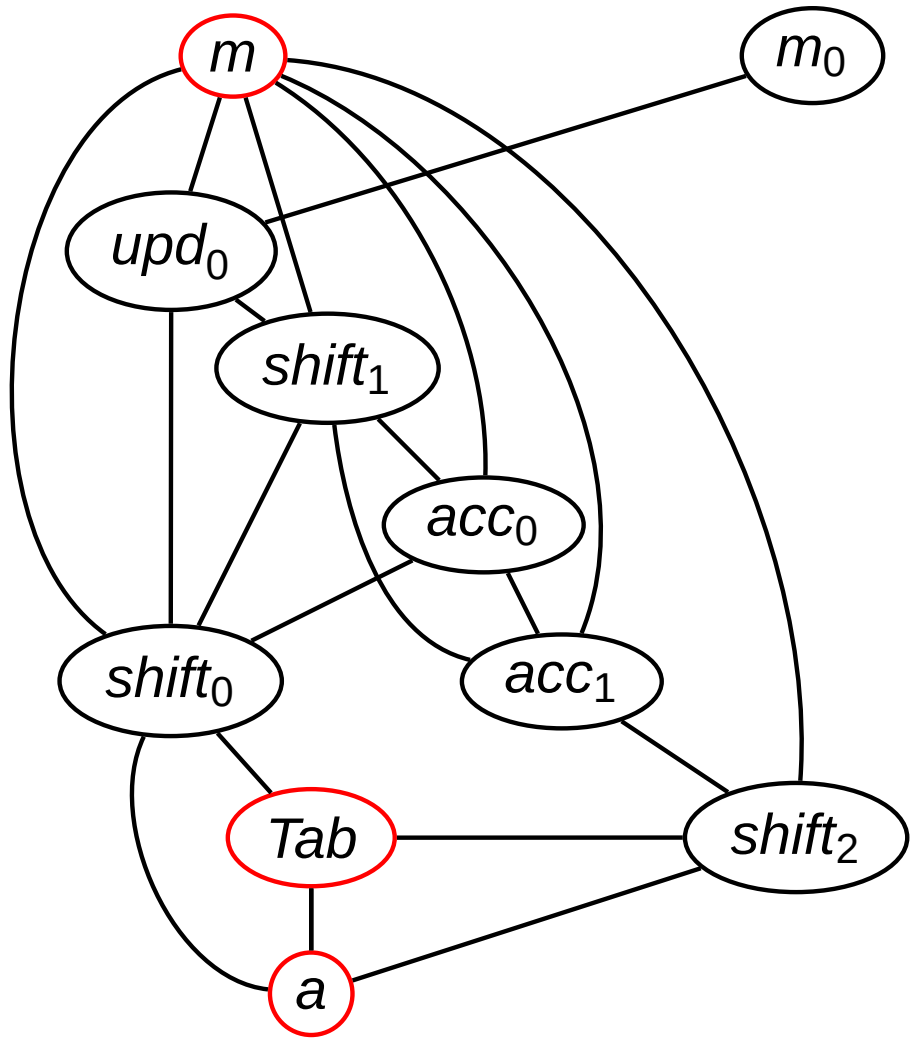


Degré de pertinence d'un terme

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$$\mathcal{T}_0 \quad = \quad \{m, m_2, Tab, a\}$$

Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$

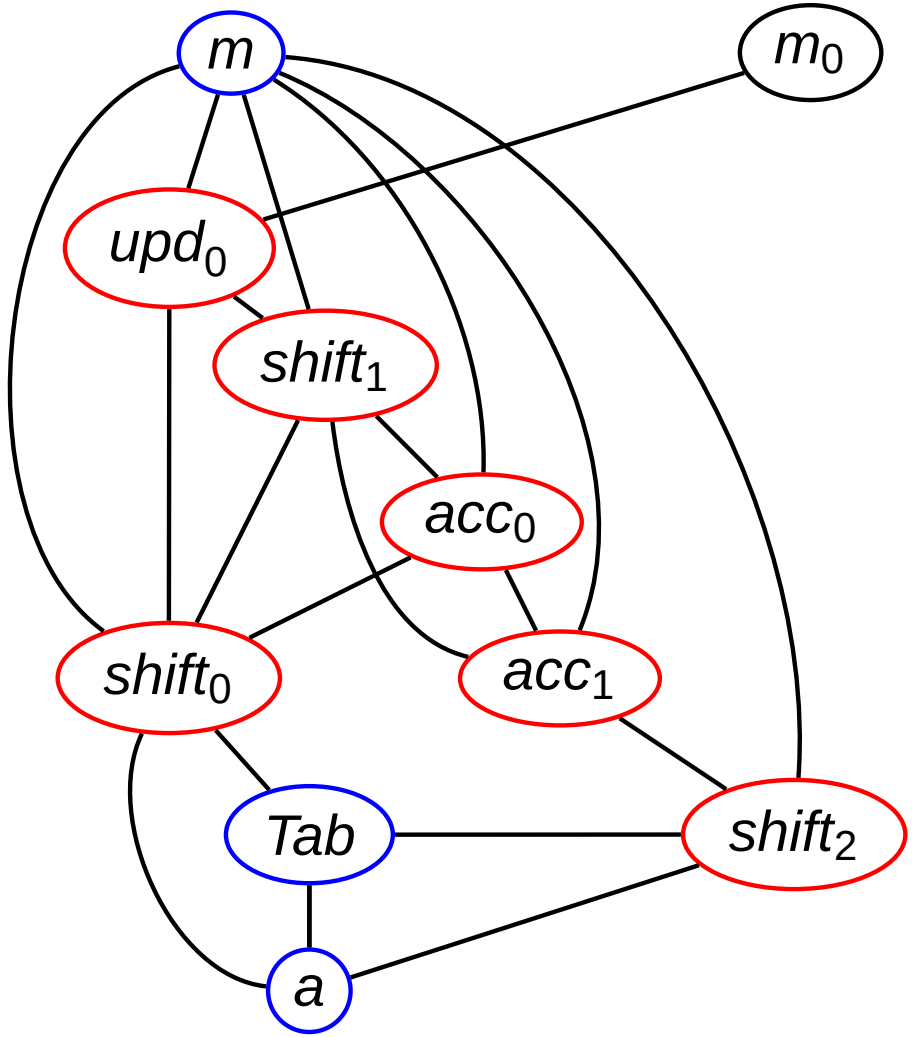


Degré de pertinence d'un terme

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$$\mathcal{T}_0 = \{m, m_2, Tab, a\}$$
$$\mathcal{T}_1 = \mathcal{T}_0 \cup \begin{Bmatrix} shift_0, shift_1, shift_2, \\ upd_0, acc_0, acc_1 \end{Bmatrix}$$

Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$

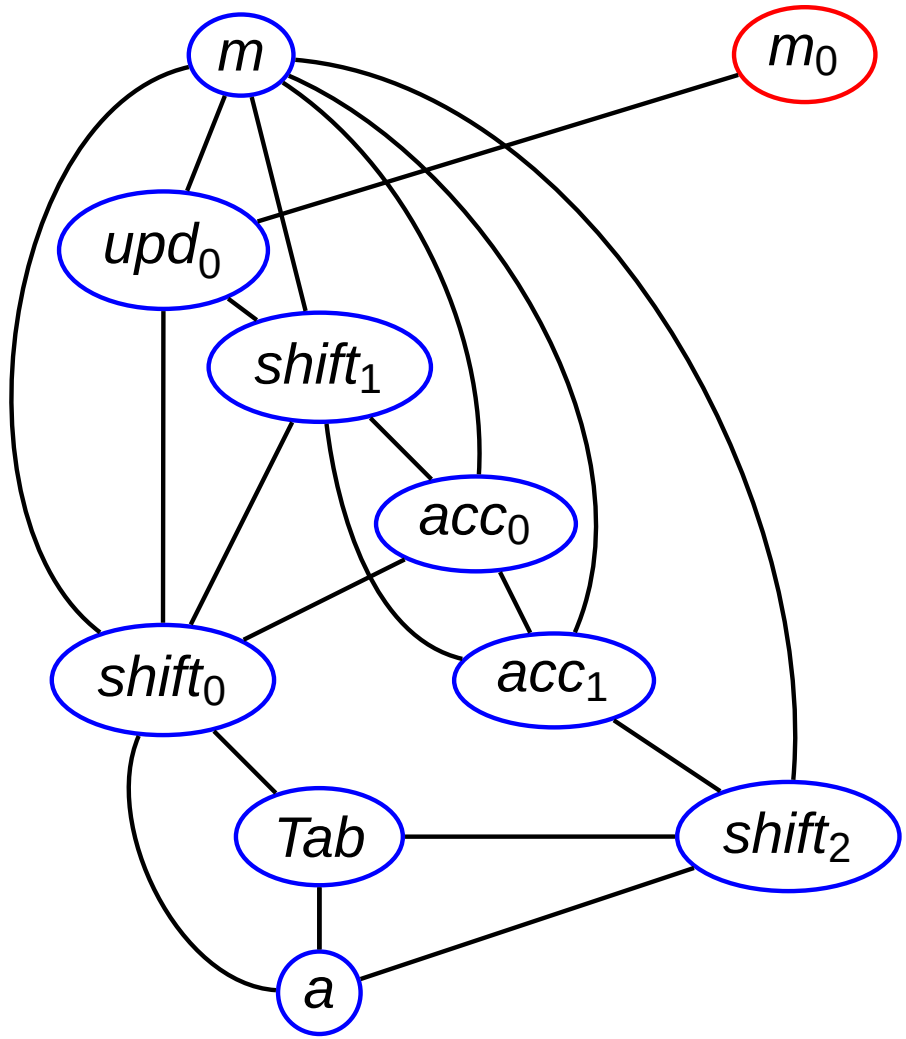


Degré de pertinence d'un terme

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$$
\begin{aligned}
\mathcal{T}_0 &= \{m, m_2, Tab, a\} \\
\mathcal{T}_1 &= \mathcal{T}_0 \cup \begin{Bmatrix} shift_0, shift_1, shift_2, \\ upd_0, acc_0, acc_1 \end{Bmatrix} \\
\mathcal{T}_2 &= \mathcal{T}_1 \cup \{m_0\}
\end{aligned}
$$

Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$

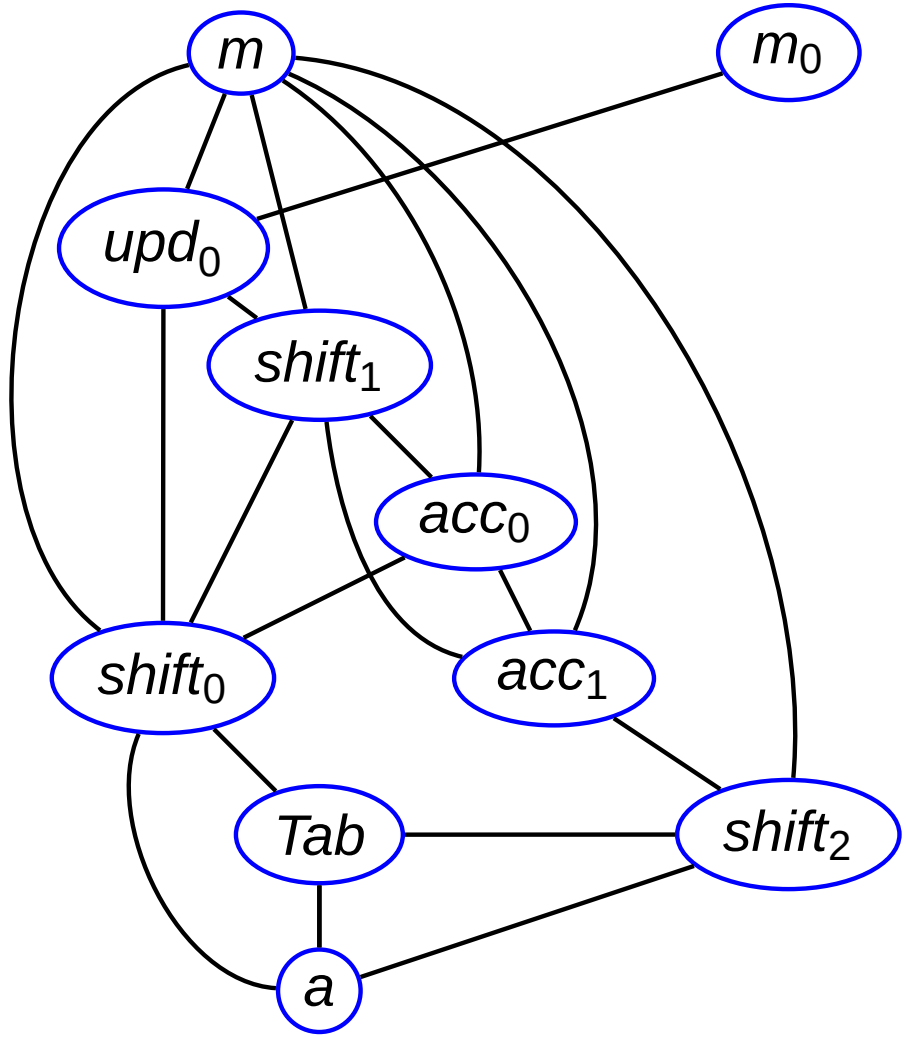


Degré de pertinence d'un terme

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$$
\begin{aligned}
\mathcal{T}_0 &= \{m, m_2, Tab, a\} \\
\mathcal{T}_1 &= \mathcal{T}_0 \cup \begin{Bmatrix} shift_0, shift_1, shift_2, \\ upd_0, acc_0, acc_1 \end{Bmatrix} \\
\mathcal{T}_2 &= \mathcal{T}_1 \cup \{m_0\} \\
&\dots
\end{aligned}
$$

Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$



Graphe de dépendance des prédicats

- Construit à partir de l'axiomatisation du langage
- Graphe orienté : considération de la polarité des prédicats
- Méthode proposée :
  1. Pour chaque axiome

$$\mathbf{diff}(m_1, m_2, lp) \Leftrightarrow \left(\forall p \,.\, \backslash\mathbf{valid}(p) \ \wedge\ \neg\mathbf{mem}(p, lp) \Rightarrow acc(m_1, p) = acc(m_2, p)\right)$$

  2. On le met en CNF
     (Forme normale conjonctive – p_0 issue de la skolémisation de p)

$$\begin{array}{l}
\left(\neg\mathbf{diff}(m_1, m_2, lp) \ \vee\ \neg\backslash\mathbf{valid}(p) \ \vee\ \mathbf{mem}(p, lp) \ \vee\ acc(m_1, p) = acc(m_2, p)\right) \\
\wedge\ \left(\backslash\mathbf{valid}(p_0) \ \vee\ \mathbf{diff}(m_1, m_2, lp)\right) \\
\wedge\ \left(\neg\mathbf{mem}(p_0, lp) \ \vee\ \mathbf{diff}(m_1, m_2, lp)\right) \\
\wedge\ \left(acc(m_1, p_0) \neq acc(m_2, p_0) \ \vee\ \mathbf{diff}(m_1, m_2, lp)\right)
\end{array}$$

  3. Chaque clause est caractérisée par ses prédicats

Graphe de dépendance des prédicats

- Construit à partir de l'axiomatisation du langage
- Graphe orienté : considération de la polarité des prédicats
- Méthode proposée :
  1. Pour chaque axiome

$$\mathbf{diff}(m_1, m_2, lp) \Leftrightarrow \big(\forall p \,.\, \backslash\mathbf{valid}(p) \ \wedge\ \neg\mathbf{mem}(p, lp) \Rightarrow acc(m_1, p) = acc(m_2, p)\big)$$

  2. On le met en CNF
     (Forme normale conjonctive – p_0 issue de la skolémisation de p)

$$\begin{aligned}
&(\neg\mathbf{diff}(m_1, m_2, lp) \ \vee\ \neg\backslash\mathbf{valid}(p) \ \vee\ \mathbf{mem}(p, lp) \ \vee\ acc(m_1, p) = acc(m_2, p)) \\
\wedge\ &(\backslash\mathbf{valid}(p_0) \ \vee\ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(\neg\mathbf{mem}(p_0, lp) \ \vee\ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(acc(m_1, p_0) \neq acc(m_2, p_0) \ \vee\ \mathbf{diff}(m_1, m_2, lp))
\end{aligned}$$

  3. Chaque clause est caractérisée par ses prédicats

Graphe de dépendance des prédicats

- Construit à partir de l'axiomatisation du langage
- Graphe orienté : considération de la polarité des prédicats
- Méthode proposée :
  1. Pour chaque axiome

$$\mathbf{diff}(m_1, m_2, lp) \Leftrightarrow \big(\forall p \,.\, \backslash\mathbf{valid}(p) \ \wedge \ \neg\mathbf{mem}(p, lp) \Rightarrow acc(m_1, p) = acc(m_2, p)\big)$$

  2. On le met en CNF
     (Forme normale conjonctive – p_0 issue de la skolémisation de p)

$$\begin{aligned}
&(\neg\mathbf{diff}(m_1, m_2, lp) \ \vee \ \neg\backslash\mathbf{valid}(p) \ \vee \ \mathbf{mem}(p, lp) \ \vee \ acc(m_1, p) = acc(m_2, p)) \\
\wedge\ &(\backslash\mathbf{valid}(p_0) \ \vee \ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(\neg\mathbf{mem}(p_0, lp) \ \vee \ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(acc(m_1, p_0) \neq acc(m_2, p_0) \ \vee \ \mathbf{diff}(m_1, m_2, lp))
\end{aligned}$$

  3. Chaque clause est caractérisée par ses prédicats

Graphe de dépendance des prédicats

- Construit à partir de l'axiomatisation du langage
- Graphe orienté : considération de la polarité des prédicats
- Méthode proposée :
  1. Pour chaque axiome

$$\mathbf{diff}(m_1, m_2, lp) \Leftrightarrow \big(\forall p \,.\, \backslash\mathbf{valid}(p) \ \wedge \ \neg\mathbf{mem}(p, lp) \Rightarrow acc(m_1, p) = acc(m_2, p)\big)$$

  2. On le met en CNF
     (Forme normale conjonctive – p_0 issue de la skolémisation de p)

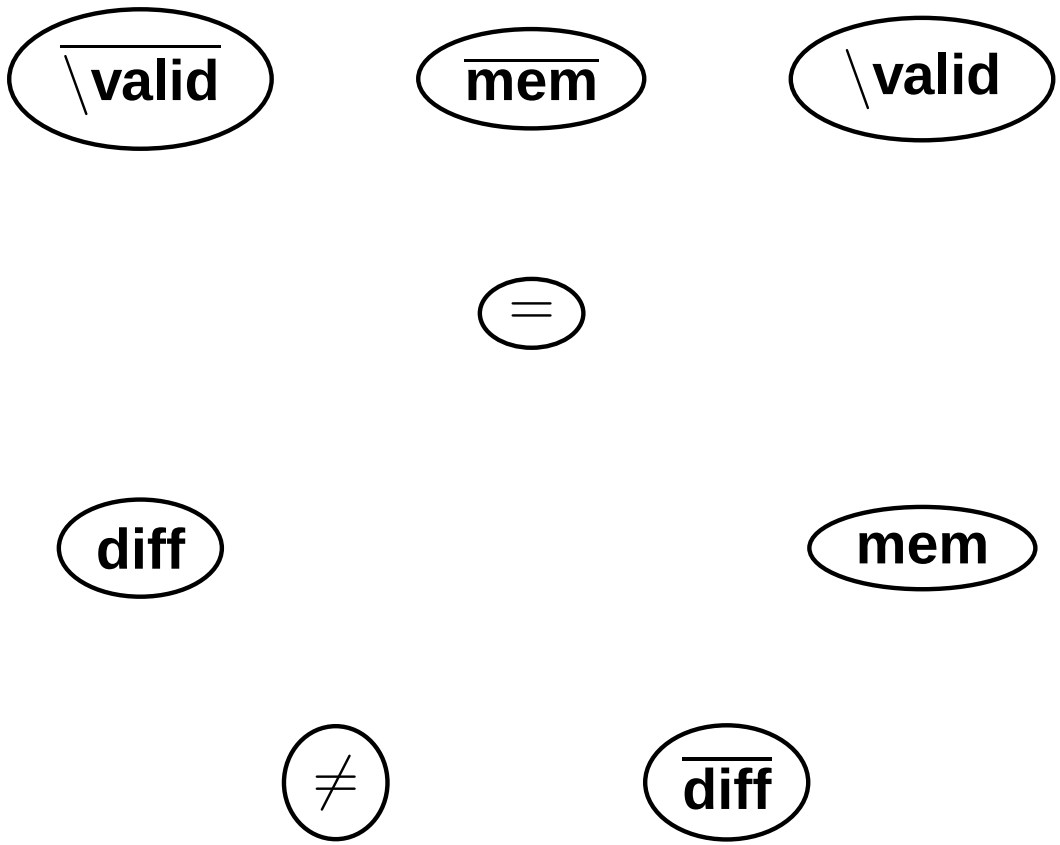
$$\begin{aligned}
&(\neg\mathbf{diff}(m_1, m_2, lp) \ \vee \ \neg\backslash\mathbf{valid}(p) \ \vee \ \mathbf{mem}(p, lp) \ \vee \ acc(m_1, p) = acc(m_2, p)) \\
\wedge\ &(\backslash\mathbf{valid}(p_0) \ \vee \ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(\neg\mathbf{mem}(p_0, lp) \ \vee \ \mathbf{diff}(m_1, m_2, lp)) \\
\wedge\ &(acc(m_1, p_0) \neq acc(m_2, p_0) \ \vee \ \mathbf{diff}(m_1, m_2, lp))
\end{aligned}$$

  3. Chaque clause est caractérisée par ses prédicats

Graphe de dépendance des prédicats

4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté *(modulo les symétries)*

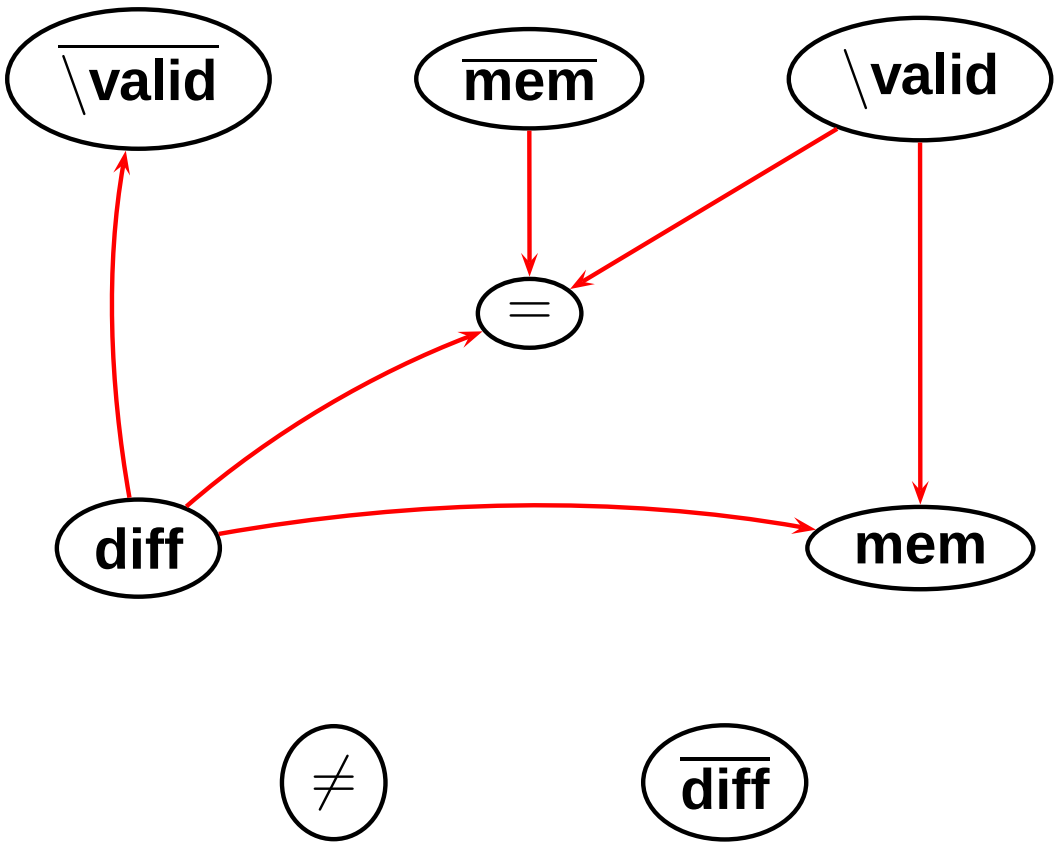
- $\{\neg\mathbf{diff}, \neg\backslash\mathbf{valid}, \mathbf{mem}, =\}$
- $\{\backslash\mathbf{valid}, \mathbf{diff}\}$
- $\{\neg\mathbf{mem}, \mathbf{diff}\}$
- $\{\neq, \mathbf{diff}\}$



Graphe de dépendance des prédicats

4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté
   (modulo les symétries)

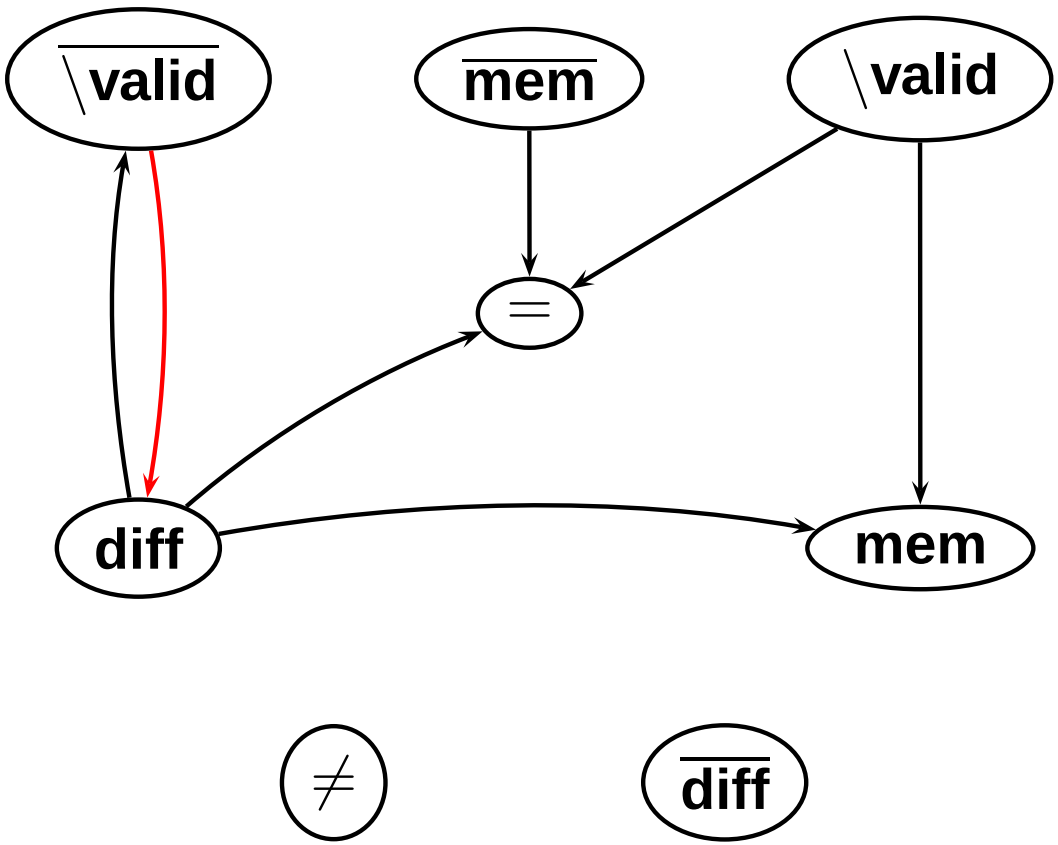
- $\{\neg\mathbf{diff}, \neg\backslash\mathbf{valid}, \mathbf{mem}, =\}$
- $\{\backslash\mathbf{valid}, \mathbf{diff}\}$
- $\{\neg\mathbf{mem}, \mathbf{diff}\}$
- $\{\neq, \mathbf{diff}\}$



Graphe de dépendance des prédicats

4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté
   (modulo les symétries)

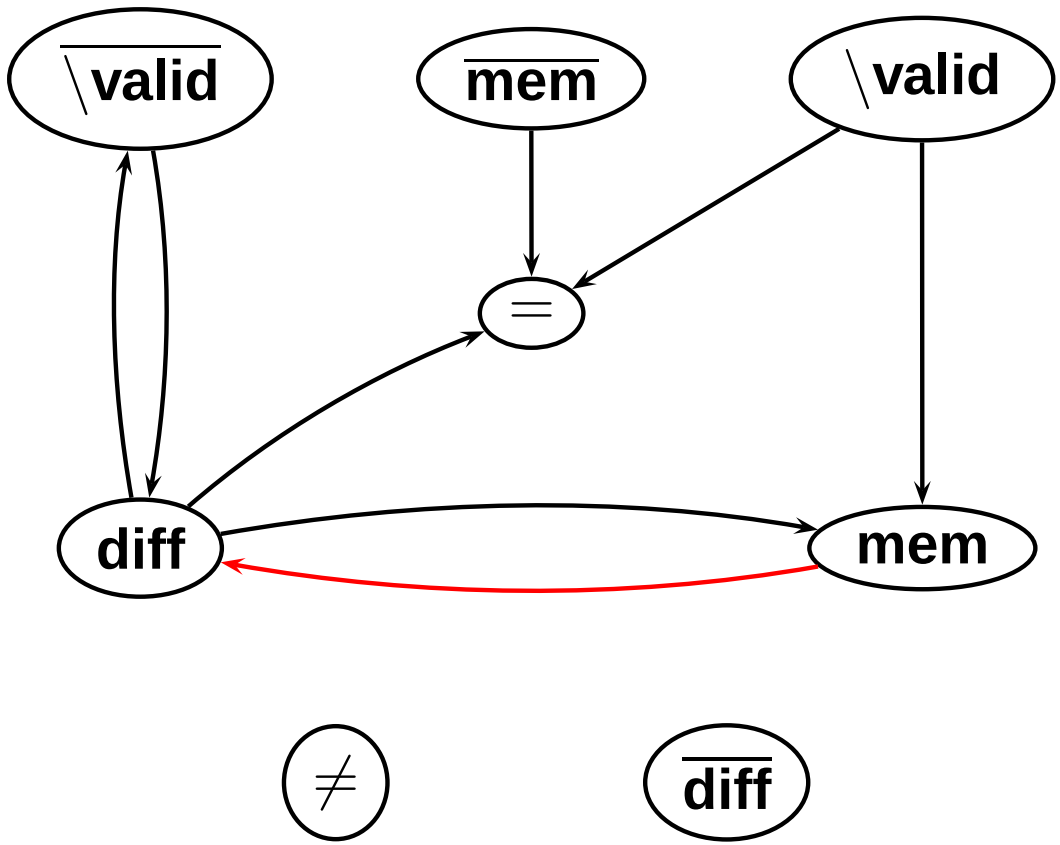
- $\{\neg\mathbf{diff}, \neg\backslash\mathbf{valid}, \mathbf{mem}, =\}$
- $\{\backslash\mathbf{valid}, \mathbf{diff}\}$
- $\{\neg\mathbf{mem}, \mathbf{diff}\}$
- $\{\neq, \mathbf{diff}\}$



Graphe de dépendance des prédicats

4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté
   (modulo les symétries)

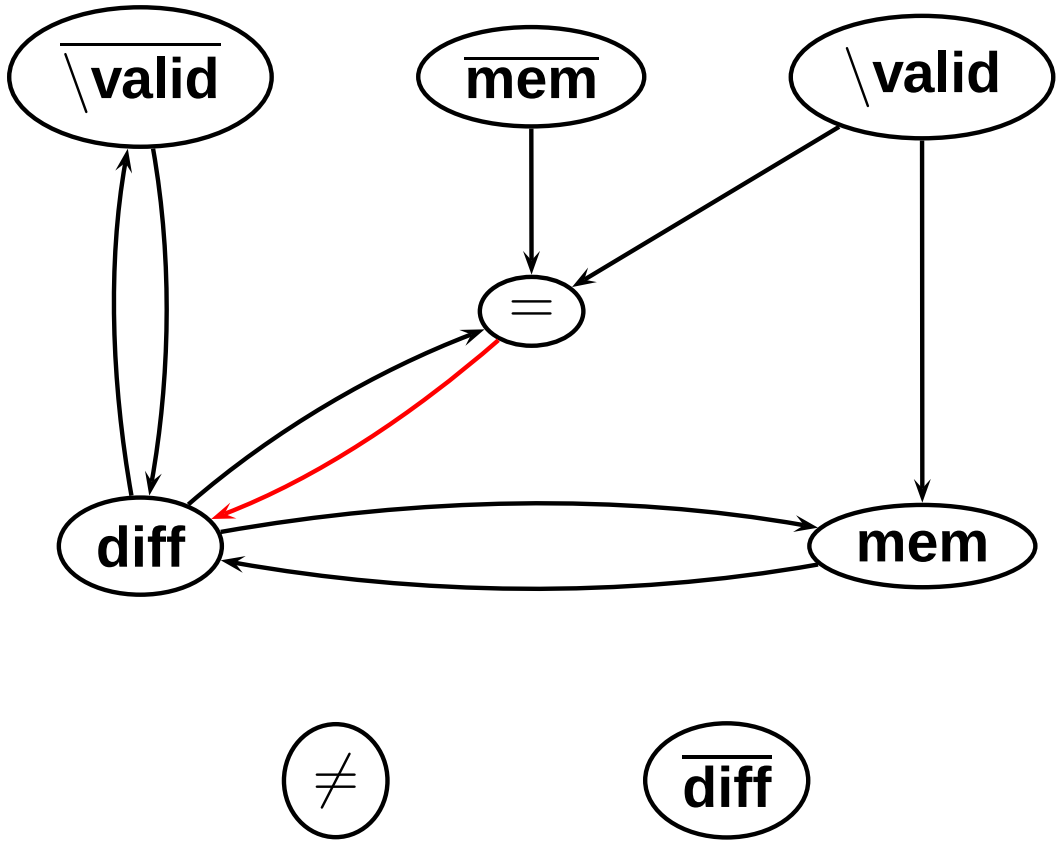
- $\{\neg\textbf{diff}, \neg\backslash\textbf{valid}, \textbf{mem}, =\}$
- $\{\backslash\textbf{valid}, \textbf{diff}\}$
- $\{\neg\textbf{mem}, \textbf{diff}\}$
- $\{\neq, \textbf{diff}\}$



Graphe de dépendance des prédicats

4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté
   (modulo les symétries)

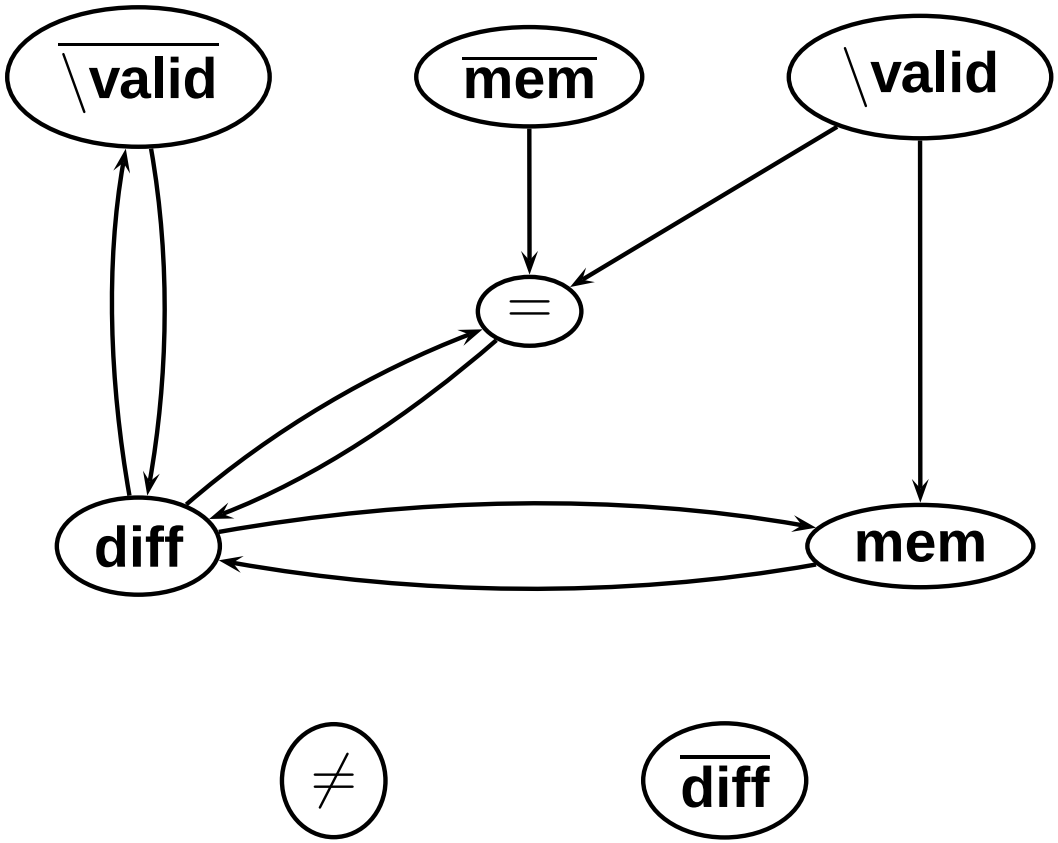
- $\{\neg\mathbf{diff}, \neg\backslash\mathbf{valid}, \mathbf{mem}, =\}$
- $\{\backslash\mathbf{valid}, \mathbf{diff}\}$
- $\{\neg\mathbf{mem}, \mathbf{diff}\}$
- $\{\neq, \mathbf{diff}\}$



Graphe de dépendance des prédicats

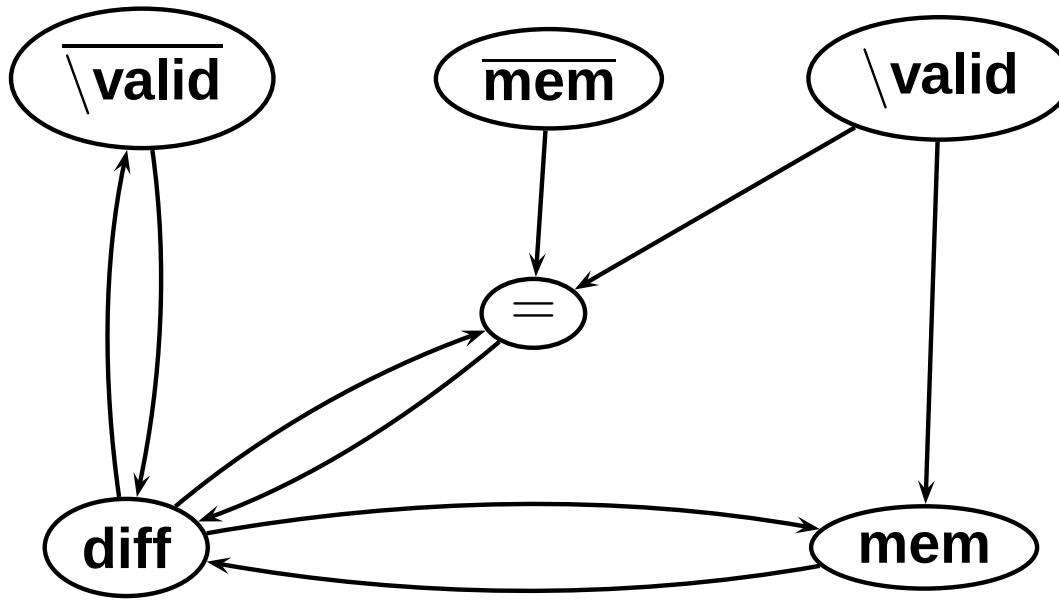
4. Chaque prédicat p devient deux noeuds : p et $\overline{p}$
5. Chaque ensemble de prédicats est fortement connecté
   (modulo les symétries)

- $\{\neg\textbf{diff}, \neg\backslash\textbf{valid}, \textbf{mem}, =\}$
- $\{\backslash\textbf{valid}, \textbf{diff}\}$
- $\{\neg\textbf{mem}, \textbf{diff}\}$
- $\{\neq, \textbf{diff}\}$



Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

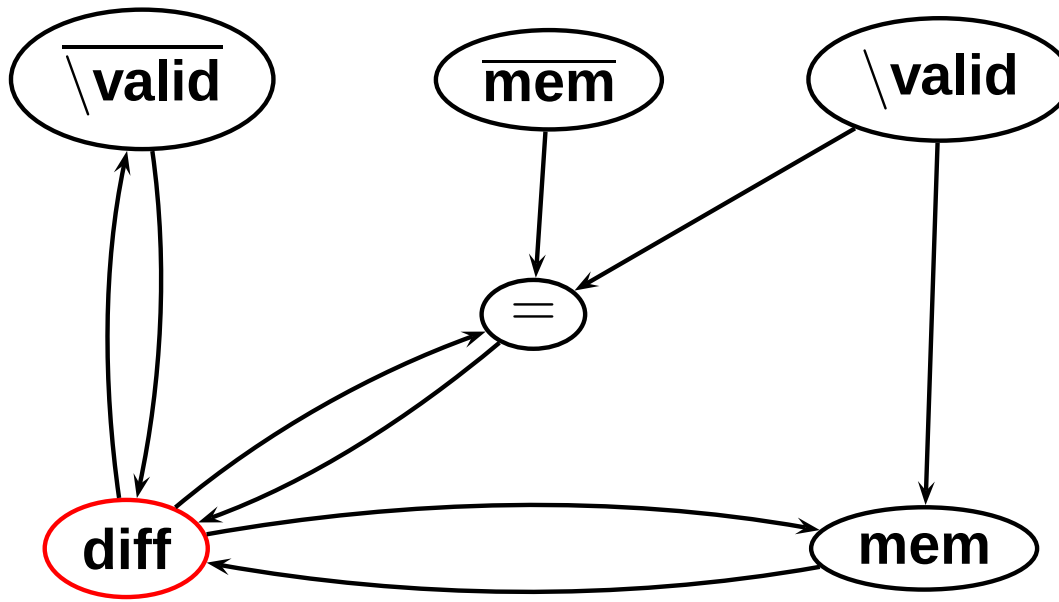


Rappel du but :
diff$(m, m_2, \{Tab+a+1, Tab+a\})$

Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$\mathcal{P}_0 = \{\textbf{diff}\}$

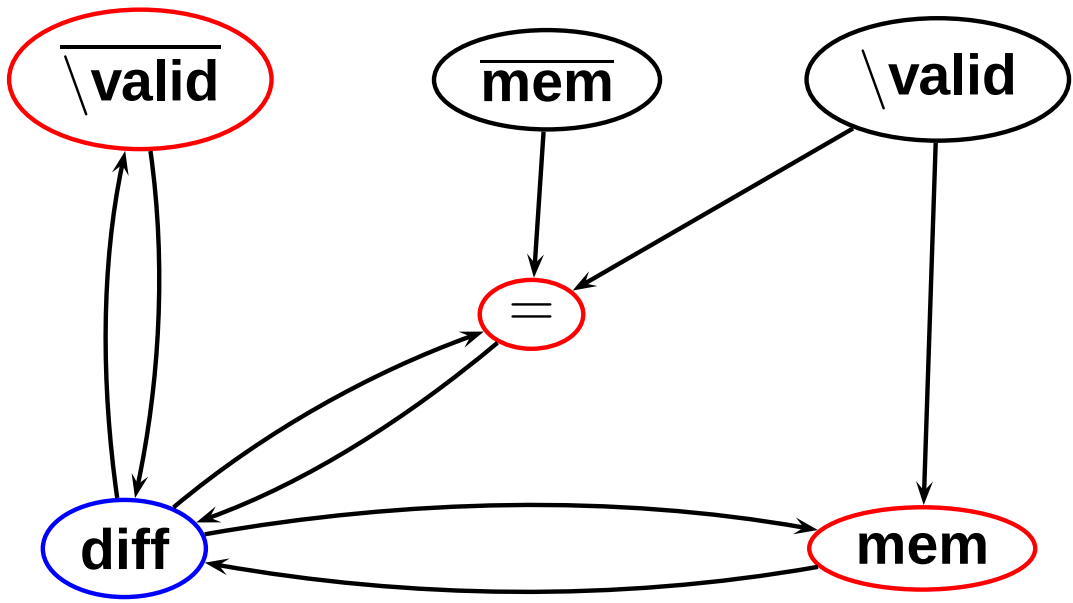


Rappel du but :
$\textbf{diff}(m, m_2, \{Tab+a+1, Tab+a\})$

Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$\mathcal{P}_0 = \{\mathbf{diff}\}$
$\mathcal{P}_1 = \mathcal{P}_0 \cup \{\overline{\backslash\mathbf{valid}}, \mathbf{mem}, =\}$

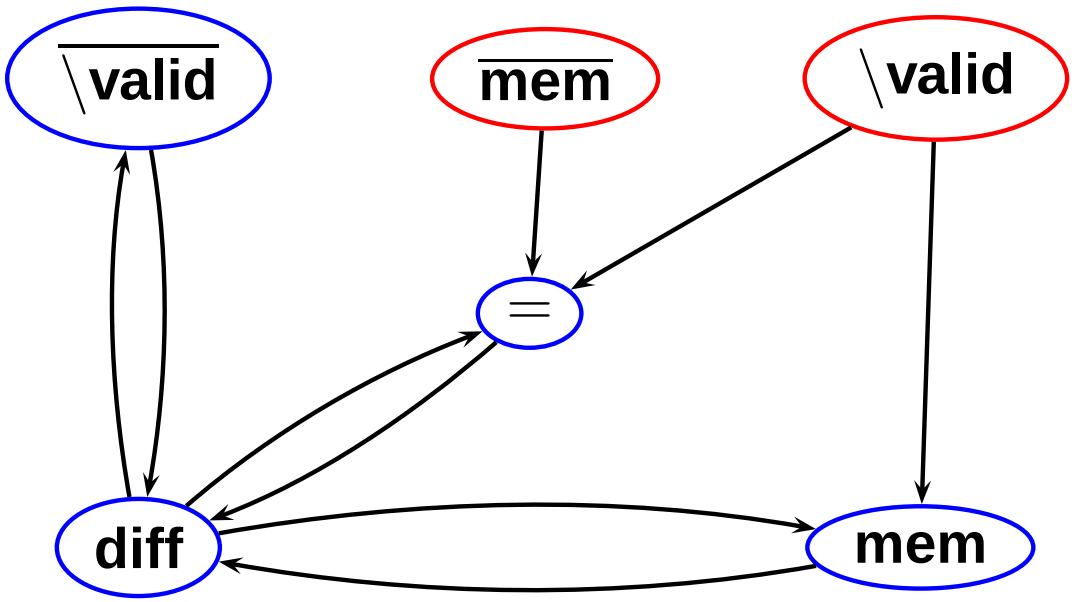


Rappel du but :
$\mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\})$

Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$\mathcal{P}_0 = \{\mathbf{diff}\}$
$\mathcal{P}_1 = \mathcal{P}_0 \cup \{\overline{\backslash\mathbf{valid}}, \mathbf{mem}, =\}$
$\mathcal{P}_2 = \mathcal{P}_1 \cup \{\overline{\mathbf{mem}}, \backslash\mathbf{valid}\}$



Rappel du but :
$\mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\})$

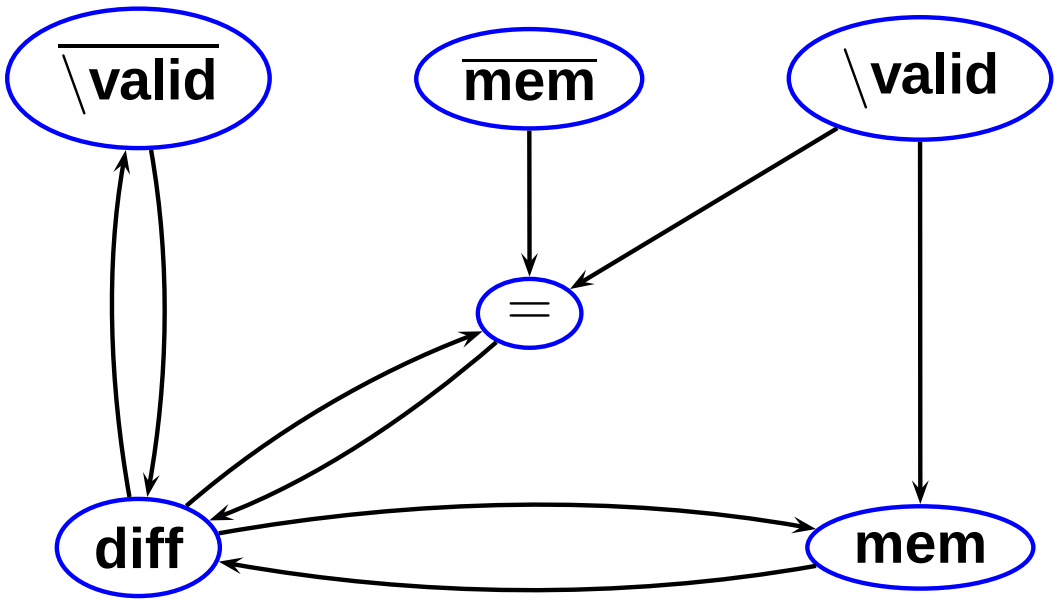
Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

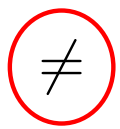
$\mathcal{P}_0 = \{\mathbf{diff}\}$
$\mathcal{P}_1 = \mathcal{P}_0 \cup \{\overline{\backslash\mathbf{valid}}, \mathbf{mem}, =\}$
$\mathcal{P}_2 = \mathcal{P}_1 \cup \{\overline{\mathbf{mem}}, \backslash\mathbf{valid}\}$
$\mathcal{P}_3 = \mathcal{P}_2 \cup \{\neq, \overline{\mathbf{diff}}\}$

Parcours modulo la symétrie :

$$(\overline{\backslash\mathbf{valid}} \implies \mathbf{diff})$$
$$\Leftrightarrow$$
$$(\overline{\mathbf{diff}} \implies \backslash\mathbf{valid})$$



Rappel du but :
$\mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\})$



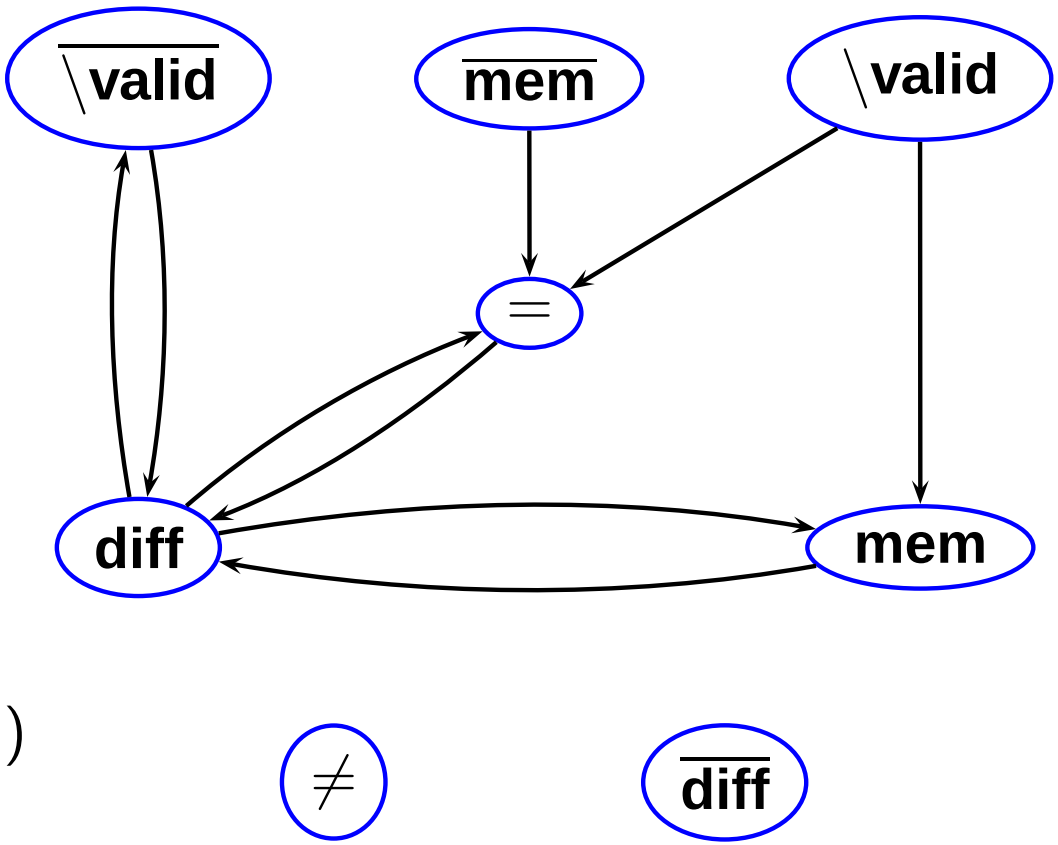
Degré de pertinence d'un prédicat

- Partant des termes de la conclusion
- Les termes les plus proches de ceux du but sont plus pertinents

$\mathcal{P}_0 = \{\mathbf{diff}\}$
$\mathcal{P}_1 = \mathcal{P}_0 \cup \{\overline{\backslash\mathbf{valid}}, \mathbf{mem}, =\}$
$\mathcal{P}_2 = \mathcal{P}_1 \cup \{\overline{\mathbf{mem}}, \backslash\mathbf{valid}\}$
$\mathcal{P}_3 = \mathcal{P}_2 \cup \{\neq, \overline{\mathbf{diff}}\}$
...

Parcours modulo la symétrie :

$$(\overline{\backslash\mathbf{valid}} \implies \mathbf{diff})$$
$$\Leftrightarrow$$
$$(\overline{\mathbf{diff}} \implies \backslash\mathbf{valid})$$



Rappel du but :
$\mathbf{diff}(m, m_2, \{Tab+a+1, Tab+a\})$

Sélection d'hypothèses

- Plusieurs critères de sélection de H possibles :
  1. Contient au moins un terme ou un prédicat pertinent

$$T \cap \mathcal{T}_{i_t} \neq \emptyset \ \vee \ P \cap \mathcal{P}_{i_p} \neq \emptyset$$

  2. Contient au moins un certain seuil de termes et prédicats

$$\text{card}(T \cap \mathcal{T}_{i_t})/\text{card}(\mathcal{T}_{i_t}) > s_t \ \vee \ \text{card}(P \cap \mathcal{P}_{i_p})/\text{card}(\mathcal{P}_{i_p}) > s_p$$

  3. Ne contient que des termes et des prédicats pertinents

$$T \subseteq \mathcal{T}_{i_t} \ \wedge \ P \subseteq \mathcal{P}_{i_p}$$

- En pratique, (1) et (2) sont trop faibles

Méthode générale

1. Construction des graphes de dépendance *(termes et prédicats)*
2. Attribution d'une pertinence à chaque terme et prédicat
3. Partant d'une sélection très restrictive :
   - Filtrage des hypothèses et preuve automatique

Ok	C'est prouvé
Timeout / Invalid	On abandonne
Unknown	On recommence avec un filtrage plus lâche

Sélection d'hypothèses

- Plusieurs critères de sélection de H possibles :
  1. Contient au moins un terme ou un prédicat pertinent

$$T \cap \mathcal{T}_{i_t} \neq \emptyset \ \vee \ P \cap \mathcal{P}_{i_p} \neq \emptyset$$

  2. Contient au moins un certain seuil de termes et prédicats

$$\mathrm{card}(T \cap \mathcal{T}_{i_t})/\mathrm{card}(\mathcal{T}_{i_t}) > s_t \ \vee \ \mathrm{card}(P \cap \mathcal{P}_{i_p})/\mathrm{card}(\mathcal{P}_{i_p}) > s_p$$

  3. Ne contient que des termes et des prédicats pertinents

$$T \subseteq \mathcal{T}_{i_t} \ \wedge \ P \subseteq \mathcal{P}_{i_p}$$

- En pratique, (1) et (2) sont trop faibles

Méthode générale

1. Construction des graphes de dépendance *(termes et prédicats)*
2. Attribution d'une pertinence à chaque terme et prédicat
3. Partant d'une sélection très restrictive :
   - Filtrage des hypothèses et preuve automatique

Ok	C'est prouvé
Timeout / Invalid	On abandonne
Unknown	On recommence avec un filtrage plus lâche

Sélection d'hypothèses (Exemple)

$$\mathcal{T}_0 = \{m, m_2, Tab, a\} \qquad \mathcal{P}_0 = \{\mathbf{diff}\}$$
$$\mathcal{T}_1 = \mathcal{T}_0 \cup \begin{Bmatrix} shift_0, shift_1, shift_2, \\ upd_0, acc_0, acc_1 \end{Bmatrix} \qquad \mathcal{P}_1 = \mathcal{P}_0 \cup \{\overline{\backslash \mathbf{valid}}, \mathbf{mem}, =\}$$
$$\mathcal{T}_2 = \mathcal{T}_1 \cup \{m_0\} \qquad \mathcal{P}_2 = \mathcal{P}_1 \cup \{\overline{\mathbf{mem}}, \backslash \mathbf{valid}\}$$
$$\mathcal{P}_3 = \mathcal{P}_2 \cup \{\neq, \overline{\mathbf{diff}}\}$$

- Pour sélectionner l'hypothèse :

$$m_0 = upd\left(\ m,\ \ Tab{+}a,\ \ acc(m, Tab{+}a)\ +\ acc(m, Tab{+}a{+}1)\ \right)$$

- Caractérisée par :
  $T = \{m, m_0, upd_0, shift_0, shift_1, shift_2, Tab, a, acc_0, acc_1\}$
  $P = \{=\}$

- Dans cet exemple simplifié, il faut au minimum $i_t = 2$ et $i_p = 1$

- Cette sélection permet d'établir automatiquement l'exemple

Démo de l’exemple fil rouge

Étude de cas Dassault

- Extrait du programme de contrôle d'avions civils
- 4000 lignes de code C
- 184000 conditions de vérification
- dont 65 non vérifiées automatiquement
  - Finalement déchargées grâce à l'approche proposée
  - avec :
    - $i_t \leq 4$
    - $i_p \leq 1$
    - timeout = 240s
- Particularités de cet exemple :
  - Hypothèses très grosses (200 littéraux)
  - Aucune hypothèse n'est quantifiée

Étude de cas Dassault

- Extrait du programme de contrôle d'avions civils
- 4000 lignes de code C
- 184000 conditions de vérification
- dont 65 non vérifiées automatiquement
  - Finalement déchargées grâce à l'approche proposée
  - avec :
    - $i_t \leq 4$
    - $i_p \leq 1$
    - timeout = 240s
- Particularités de cet exemple :
  - Hypothèses très grosses (200 littéraux)
  - Aucune hypothèse n'est quantifiée

Étude de cas Oslo

- Oslo est un Open Secure LOader :
  - Logiciel de bas niveau utilisant le TPM
  - Lanceur de programmes
  - Construit un environnement d'exécution de confiance
- Les hypothèses sont nombreuses et « fortement » quantifiées
- 1500 lignes de code C et assembleur
- 7300 CVs
- Extrait étudié :
  - 40 SLOC + 500 lignes de spécification
  - 771 CVs générées
  - 707 CVs déchargées automatiquement par Simplify
  - 42 supplémentaires grâce à la sélection d'hypothèses

Bilan

- Méthode de sélection des hypothèses basée sur les graphes
  1. Construction des graphes de dépendance *(termes et prédicats)*
  2. Calcul des ensembles de termes et prédicats pertinents
  3. Filtrage des hypothèses par rapport à ces ensembles
  4. Méthode incrémentale jusqu'à divergence du prouveur

- Le code développé est intégré à l'outil WHY
  - Langage OCaml
  - Moins de 2000 lignes de code
  - Heuristiques sélectionnables par ligne de commande
  - Disponible en téléchargement libre

- La CV de l'exemple fil rouge se décharge en 0.40 secondes
  (au lieu d'un timeout après 1 minute)

Bilan et perspectives des heuristiques plus fines

- Déjà fait :
  - Mise en CNF des hypothèses
    Filtrage des clauses et non pas des formules
  - Annotation des comparaisons par les termes attenants
    $f(x) = g(y)$ *donnent les prédicats* $=_f$ *et* $=_g$
- A faire :
  - Abstraction des prédicats de comparaison
    Laisse les déductions arithmétiques aux prouveurs
  - Élaguer les hypothèses en préservant leur forme
    Limite les instantiations répétitives du prouveur
- Objectif visé : *Vérification 100% automatique d'Oslo*

Merci de votre attention

Avez vous des questions ?

Brève introduction syntaxe + calcul de WP

Syntaxe

$$\left.\begin{array}{l} //@\ \textbf{requires}\ R \quad \textit{// Pré-condition} \\ //@\ \textbf{ensures}\ Q \quad \textit{// Post-condition} \end{array}\right\} \textit{formules logiques}$$

$F\,\{\ldots\}$ *// Fonction C*

Principe de la logique de Hoare

- Si R est vrai alors l'exécution de F vérifie Q

Propriété *Tout r t.q. $r \implies R$ est aussi une pré-condition pour F et Q*

Notons $\mathit{Pré}(F, Q)$ l'ensemble des pré-conditions pour F et Q

WP = Weakest Precondition

- P : Plus faible pré-condition pour F et Q signifie que
  - $P \in \mathrm{Pré}(F, Q)$
  - $\forall r \cdot r \in \mathrm{Pré}(F, Q) \implies (r \implies P)$
- Calcul de WP : permet de construire P à partir de F et Q

Vérification *Consiste à montrer que $R \implies \mathcal{WP}(F, Q)$*

Brève introduction syntaxe + calcul de WP

Syntaxe

$$\left.\begin{array}{l} //@\ \textbf{requires}\ R\ \ \textit{// Pré-condition} \\ //@\ \textbf{ensures}\ Q\ \ \textit{// Post-condition} \end{array}\right\}\ \textit{formules logiques}$$

$F\,\{\ldots\}$ *// Fonction C*

Principe de la logique de Hoare

- Si R est vrai alors l'exécution de F vérifie Q

Propriété *Tout r t.q. $r \implies R$ est aussi une pré-condition pour F et Q*

Notons $\mathit{Pré}(F, Q)$ l'ensemble des pré-conditions pour F et Q

WP = Weakest Precondition

- P : Plus faible pré-condition pour F et Q signifie que
  - $P \in \mathrm{Pré}(F, Q)$
  - $\forall r \cdot r \in \mathrm{Pré}(F, Q) \implies (r \implies P)$
- Calcul de WP : permet de construire P à partir de F et Q

Vérification *Consiste à montrer que $R \implies \mathcal{WP}(F, Q)$*

Brève introduction syntaxe + calcul de WP

Syntaxe

$$\left.\begin{array}{l} //@\ \textbf{requires}\ R \quad \textit{// Pré-condition} \\ //@\ \textbf{ensures}\ Q \quad \textit{// Post-condition} \end{array}\right\} \textit{formules logiques}$$

$F\,\{\ldots\}$ *// Fonction C*

Principe de la logique de Hoare

- Si R est vrai alors l'exécution de F vérifie Q

Propriété *Tout r t.q. $r \implies R$ est aussi une pré-condition pour F et Q*

Notons $Pré(F, Q)$ l'ensemble des pré-conditions pour F et Q

WP = Weakest Precondition

- P : Plus faible pré-condition pour F et Q signifie que
  - $P \in \mathrm{Pré}(F, Q)$
  - $\forall r \cdot r \in \mathrm{Pré}(F, Q) \implies (r \implies P)$
- Calcul de WP : permet de construire P à partir de F et Q

Vérification *Consiste à montrer que $R \implies \mathcal{WP}(F, Q)$*

Introduction par l'exemple

//@ **requires** $x \geq 0$ // *Pré-condition*
//@ **ensures** $x > 0$ // *Post-condition*

void inc_x(){
  x=x+1 ;
  y=x ;
}

Exemple de condition de vérification générée

Préservation de la post-condition

$$\overbrace{x \geq 0}^{\textit{Pré-condition}} \implies \overbrace{(x_0 = x + 1 \ \wedge \ y_0 = x_0 \implies x_0 > 0)}^{\mathcal{WP}(inc_x, x > 0)}$$

Où x est la variable avant l'affectation et x_0 celle après l'affectation

Introduction par l'exemple

//@ **requires** $x \geq 0$ // *Pré-condition*
//@ **ensures** $x > 0$ // *Post-condition*
//@ **assigns** x // *Liste des variables modifiées*

void inc_x(){
  x=x+1 ;
  y=x ;
}

Exemple de condition de vérification générée

Pas de modification des variables autres que x

$$\overbrace{x \geq 0}^{\textit{Pré-condition}} \implies \overbrace{(x_0 = x + 1 \ \wedge \ y_0 = x_0 \implies y_0 = y)}^{\mathcal{WP}(inc_x, y_0 = y)}$$

Ce qui est faux.