Structuring the Synthesis of Heap-Manipulating Programs

NADIA POLIKARPOVA, UCSD, USA
ILYA SERGEY, University College London, UK

Introduction

$\{x \mapsto a * y \mapsto b\}$ `void swap(loc x, loc y)` $\{x \mapsto b * y \mapsto a\}$

Intérêt : Faire avancer l'état de l'art en matiére de synthèse de programmes qui manipulent des pointeurs à partir de spécifications fonctionelles formelles.
Idée Clé : Utiliser la logique de séparation.
Contributions : Synthetic Separation Logic un systeme de preuve. Et SuSLik leur synthétiseur

Spécifications pour la Synthèse

On utilise ici des tas symboliques.

$$\Sigma; \Gamma; \{\mathcal{P}\} \rightsquigarrow \{\mathcal{Q}\}|c$$

- Γ : environnement
- Σ : contexte
- $\mathcal{P}$,ϕ,P : précondition, ses parties pure et spatiale
- $\mathcal{Q}$,ψ,Q : postcondition, ses parties pure et spatiale
- $GV(\Gamma, \mathcal{P}, \mathcal{Q}) = Vars(\mathcal{P}) \backslash \Gamma$
- $EV(\Gamma, \mathcal{P}, \mathcal{Q}) = Vars(\mathcal{Q}) \backslash (\Gamma \cup Vars(\mathcal{P})$

Règles d'Inférence Basiques

Un exemple

$$\textsc{Emp}\quad \frac{\mathsf{EV}(\Gamma, \mathcal{P}, \mathcal{Q}) = \emptyset \qquad \phi \Rightarrow \psi}{\Gamma;\ \{\phi; \mathsf{emp}\} \rightsquigarrow \{\psi; \mathsf{emp}\} \mid \mathsf{skip}}$$

$$\textsc{Read}\quad \frac{a \in \mathsf{GV}(\Gamma, \mathcal{P}, \mathcal{Q}) \qquad y \notin \mathsf{Vars}(\Gamma, \mathcal{P}, \mathcal{Q}) \qquad \Gamma \cup \{y\};\ [y/a]\{\phi; \langle x, \iota\rangle \mapsto a * P\} \rightsquigarrow [y/a]\{Q\} \mid c}{\Gamma;\ \{\phi; \langle x, \iota\rangle \mapsto a * P\} \rightsquigarrow \{Q\} \mid \mathbf{let}\ y = *(x + \iota); c}$$

$$\textsc{Write}\quad \frac{\mathsf{Vars}(e) \subseteq \Gamma \qquad \Gamma;\ \{\phi; \langle x, \iota\rangle \mapsto e * P\} \rightsquigarrow \{\psi; \langle x, \iota\rangle \mapsto e * Q\} \mid c}{\Gamma;\ \left\{\phi; \langle x, \iota\rangle \mapsto e' * P\right\} \rightsquigarrow \left\{\psi; \langle x, \iota\rangle \mapsto e * Q\right\} \ \middle|\ *(x + \iota) = e; c}$$

$$\textsc{Frame}\quad \frac{\mathsf{EV}(\Gamma, \mathcal{P}, \mathcal{Q}) \cap \mathsf{Vars}(R) = \emptyset \qquad \Gamma;\ \{\phi; P\} \rightsquigarrow \{\psi; Q\} \mid c}{\Gamma;\ \{\phi; P * R\} \rightsquigarrow \{\psi; Q * R\} \mid c}$$

Fig. 1. Simplified basic rules of SSL.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \{\mathsf{emp}\} \rightsquigarrow \{\mathsf{emp}\}}\ \textsc{Emp with } c_7 = \mathsf{skip}}{c_6 = c_7}}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{y \mapsto \mathsf{a2}\right\} \rightsquigarrow \left\{y \mapsto \mathsf{a2}\right\} \mid c_6}\ \textsc{Frame}}{c_5 = *y = \mathsf{a2}; c_6}}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{y \mapsto \mathsf{b2}\right\} \rightsquigarrow \left\{y \mapsto \mathsf{a2}\right\} \mid c_5}\ \textsc{Write}}{c_4 = c_5}}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{x \mapsto \mathsf{b2} * y \mapsto \mathsf{b2}\right\} \rightsquigarrow \left\{x \mapsto \mathsf{b2} * y \mapsto \mathsf{a2}\right\} \mid c_4}\ \textsc{Frame}}{c_3 = *x = \mathsf{b2}; c_4}}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{x \mapsto \mathsf{a2} * y \mapsto \mathsf{b2}\right\} \rightsquigarrow \left\{x \mapsto \mathsf{b2} * y \mapsto \mathsf{a2}\right\} \mid c_3}\ \textsc{Write}}{c_2 = \mathbf{let}\ \mathsf{b2} = *y; c_3}}{\{x, y, \mathsf{a2}\};\ \left\{x \mapsto \mathsf{a2} * y \mapsto b\right\} \rightsquigarrow \{x \mapsto b * y \mapsto \mathsf{a2}\} \mid c_2}\ \textsc{Read}}{c_1 = \mathbf{let}\ \mathsf{a2} = *x; c_2}}{\{x, y\};\ \left\{x \mapsto a * y \mapsto b\right\} \rightsquigarrow \{x \mapsto b * y \mapsto a\} \mid c_1}\ \textsc{Read}$$

Fig. 2. Derivation of swap(x,y) as c_1.

Règles d'Inférence Basiques

EMP terminale, parties spatiales vide, $EV = \emptyset$, $\phi \implies \psi$
skip

READ assigne la valeur d'une GV a une nouvelle variable de programme et substitue toutes les occurences.
let b = *x

WRITE assigne l'évaluation d'une expression e à une case mémoire.
*x = b

FRAME Enlève une partie spatiale commune à ϕ et ψ, si cela ne crée pas de variable existentielle.
skip

Unification Spatiale et Backtrack

Raisonner sur les contraintes pures

Préconditions

Raisonner sur les contraintes pures

Postconditions

Synthèse pour prédicats inductifs

Mémoire dynamique

Synthèse pour prédicats inductifs

Induction

Synthèse pour prédicats inductifs

Déroulement de prédicat

Synthèse pour prédicats inductifs

Etiquette de niveau

Synthèse pour prédicats inductifs

Déroulement dans la postcondition

Permettre l'appel de procèdure

Enlévement de l'appel

Synthetic Separation Logic

Garanties Formelles

Algorithme de synthèse basé sur SSL

Optimisations et extensions

Optimisations :

- Règles inversibles

Optimisations et extensions

Optimisations :

- Règles inversibles
- Recherche multi-phase

Optimisations et extensions

Optimisations :

- Règles inversibles
- Recherche multi-phase
- Rèduction des symétries

Optimisations et extensions

Optimisations :

- Règles inversibles
- Recherche multi-phase
- Rèduction des symétries
- Règles d'échec

Optimisations et extensions

Optimisations :

- Règles inversibles
- Recherche multi-phase
- Rèduction des symétries
- Règles d'échec

Extensions :

- Fonctions auxilliaire

Optimisations et extensions

Optimisations :

- Règles inversibles
- Recherche multi-phase
- Rèduction des symétries
- Règles d'échec

Extensions :

- Fonctions auxilliaire
- Enlèvement de branches

Benchmark