Translation Validation via Linear Recursion Schemes

Master Seminar

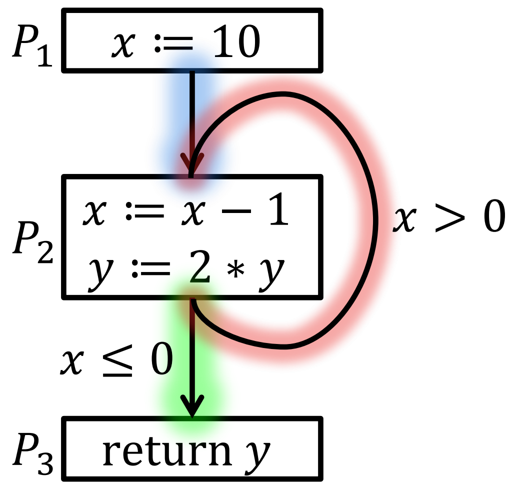
Tobias Tebbi

Translation Validation

- **Goal** Verified Compiler
- **Method** Implement Validator that checks if input and output of compiler pass are equivalent.
- **Needs** Decidable sufficient criterion for program equivalence

CPS

Control **F**low **G**raph



Continuation **P**assing **S**tyle

$$\boldsymbol{P_1(x, y) := P_2(10, y)}$$

$$\boldsymbol{P_2(x, y) :=}$$

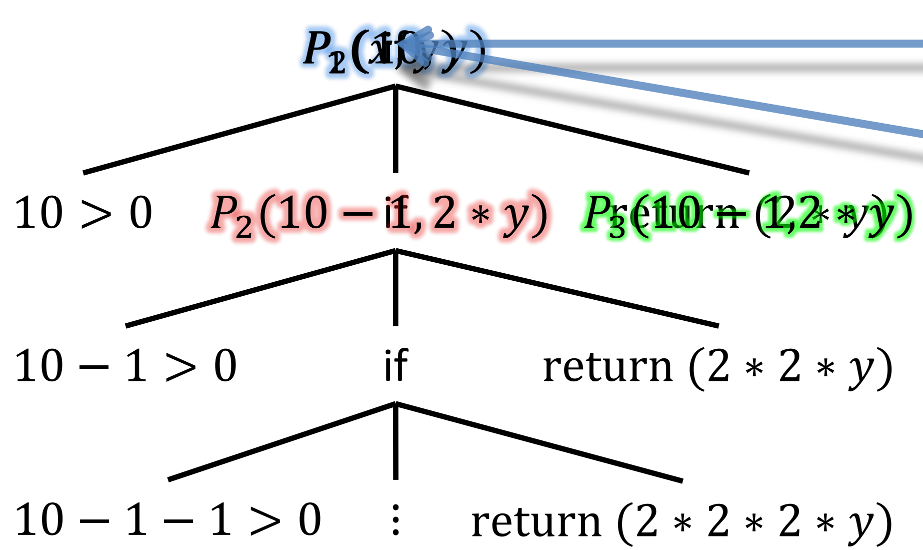
$$\textbf{if } \boldsymbol{x > 0}$$

$$\textbf{then } \boldsymbol{P_2(x - 1, 2 * y)}$$

$$\textbf{else } \boldsymbol{P_3(x - 1, 2 * y)}$$

$$\boldsymbol{P_3(x, y) :=} \textbf{ return } \boldsymbol{y}$$

Unfolding the Procedures



$P_1(x, y) \coloneqq \boldsymbol{P_2(10, y)}$

$P_2(x, y) \coloneqq$

if $\boldsymbol{x > 0}$

then $\boldsymbol{P_2(x - 1, 2 * y)}$

else $\boldsymbol{P_3(x - 1, 2 * y)}$

$P_3(x, y) \coloneqq$ **return** $\boldsymbol{y}$

Program Equivalence

- If trees equal, then programs equivalent.
- This is decidable! [Sabelfeld2000]
- Many optimizations do not change the tree.
- It does **not** matter
  - which arguments/variables/registers are used.
  - when values are computed.
- But the branching structure **does** matter, e.g. which test is done first.

Linear Recursion Scheme

- Restriction with polynomial equivalence check

interpreted **procedures**

$$P(x,y,z) \coloneqq f(g(x,y), P'(h(z)), P'(x))$$

linear: **nesting forbidden**

uninterpreted **functions**
e.g. +, <, if-then-else,
return, …

Simplifications for this Talk

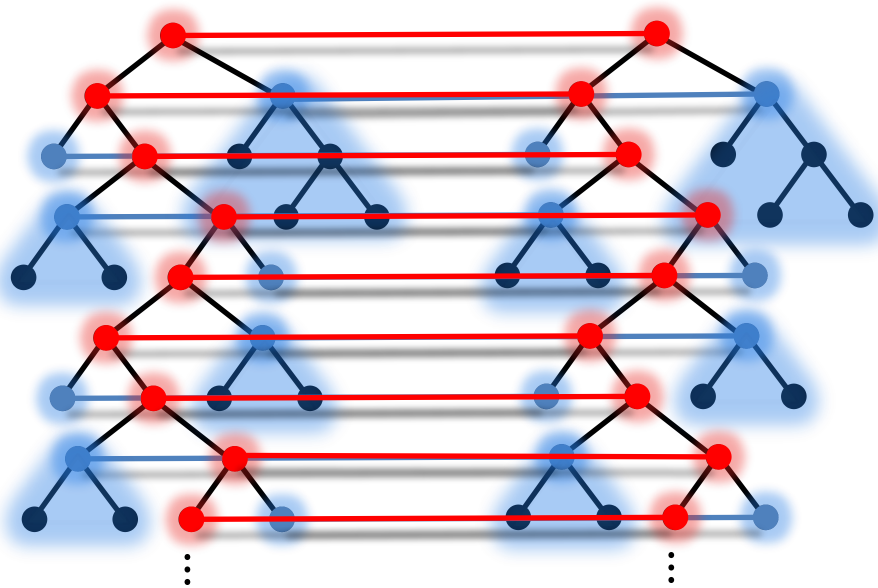
- Just one uninterpreted function/operator $s \cdot t$
- Simple terms $s, t ::= x \mid a \mid s \cdot t$
- Terms $S, T ::= s \mid P(s)$
- Only procedures of the form

$$P(x) \coloneqq P'(s) \cdot t$$
$$P(x) \coloneqq s \cdot P'(t)$$
$$P(x) \coloneqq P'(s) \cdot P''(t)$$

- Thus
  - All procedures produce infinite trees
  - Only binary trees where all inner nodes are labelled with $\cdot$ and leaves are labelled with variables or constants
  - Every subtree is described by a term $P(s)$ or s

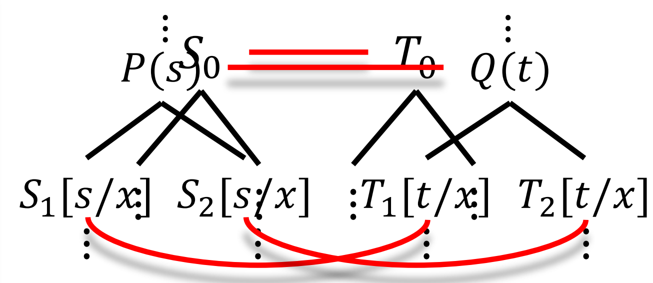
Equality of Infinite Trees

- Binary infinite trees equal ⇔ All subtrees at same position and with infinite parent-subtrees are both infinite or equal



Equality of Infinite Trees

- Binary infinite trees equal ⇔ All subtrees at same position and with infinite parent-subtrees are both infinite or equal
- To check equivalence of S_0 and T_0, we generate all such pairs of subtrees with the inductively defined relation $\mathrm{Unf}(S_0, T_0)$:
  - $(S_0, T_0) \in \mathrm{Unf}(S_0, T_0)$
  - If $\big(P(s), Q(t)\big) \in \mathrm{Unf}(S_0, T_0)$ with $P(x) \coloneqq S_1 \cdot S_2$ and $\mathrm{Q}(x) \coloneqq T_1 \cdot T_2$, then $(S_1[s/x], T_1[t/x]) \in \mathrm{Unf}(S_0, T_0)$ and $(S_2[s/x], T_2[t/x]) \in \mathrm{Unf}(S_0, T_0)$



- $\mathrm{Unf}(S_0, T_0)$ is **consistent** if for all $(S, T) \in \mathrm{Unf}(S_0, T_0)$, both S and T are procedure calls or $S = T$.
- $S_0 \equiv T_0$ iff $\mathrm{Unf}(S_0, T_0)$ is consistent.

Substitutions

- A substitution σ is a function from variables to simple terms.
- σS is the term S where every occurrence of a variable x is replaced by σx.
- The instantiation pre-order $\leq$ on terms:

$$S \leq T \quad :\Leftrightarrow \quad \exists \sigma.\; S = \sigma T$$

And on pairs of terms:

$$(S_1, S_2) \leq (T_1, T_2)$$

$$:\Leftrightarrow$$

$$\exists \sigma.\; (S_1 = \sigma T_1 \wedge S_2 = \sigma T_2)$$

Finite Equivale

- $(S_0, T_0) \in \mathrm{Unf}(S_0, T_0)$
- If $(P(s), Q(t)) \in \mathrm{Unf}(S_0, T_0)$
  with $P(x) \coloneqq S_1 \cdot S_2$ and $Q(x) \coloneqq T_1 \cdot T_2$,
  then $(S_1[s/x], T_1[t/x]) \in \mathrm{Unf}(S_0, T_0)$
  and $(S_2[s/x], T_2[t/x]) \in \mathrm{Unf}(S_0, T_0)$

- If there is a consistent superset of $\mathrm{Unf}(S_0, T_0)$, then $S_0 \equiv T_0$.
- We want to construct a finite representation of such a set to serve as an equivalence proof.
- Consider a finite, consistent relation R such that
  - $(S_0, T_0) \leq (S, T)$ for some $(S, T) \in R$
  - If $(P(s), Q(t)) \in R$
    with $P(x) \coloneqq S_1 \cdot S_2$ and $Q(x) \coloneqq T_1 \cdot T_2$,
    then for $i \in \{1,2\}$,
    $S_i[s/x] = T_i[t/x]$ is a simple term
    or $(S_i[s/x], T_i[t/x]) \leq (S, T)$ for some $(S, T) \in R$

Lemma:

$$\{(S,T) \mid (S,T) \leq (S',T') \text{ with } (S',T') \in R\} \cup \{(s,s) \mid s \text{ is a simple term}\} \supseteq \mathrm{Unf}(S_0, T_0)$$

- The constraint on R is decidable. Thus we have a method to prove equivalence.

Unificat

- $(S_0, T_0) \in \mathrm{Unf}(S_0, T_0)$
- If $\big(P(s), Q(t)\big) \in \mathrm{Unf}(S_0, T_0)$
  with $P(x) \coloneqq S_1 \cdot S_2$ and $\mathrm{Q}(x) \coloneqq T_1 \cdot T_2$,
  then $(S_1[s/x], T_1[t/x]) \in \mathrm{Unf}(S_0, T_0)$
  and $(S_2[s/x], T_2[t/x]) \in \mathrm{Unf}(S_0, T_0)$

- σ is unifier of S and T if $\sigma S \equiv \sigma T$
- We write σR for $\{(\sigma S, \sigma T) \mid (S, T) \in \mathrm{R}\}$

Lemma: $\mathrm{Unf}(\sigma S, \sigma T) = \sigma \mathrm{Unf}(S, T\ )$

- Thus, $\mathrm{Unf}(\sigma S, \sigma T)$ is consistent iff $\sigma \mathrm{Unf}(S, T\ )$ is consistent iff
  - for all simple pairs $(s, t) \in \mathrm{Unf}(S, T)$, $\sigma s = \sigma t$
  - all other pairs consist of procedure calls only
- This is a classical first-order unification problem
- Thus we have **m**ost **g**eneral **u**nifiers (MGUs) σ:

  For every other unifier τ, we have $\tau = \tau \circ \sigma$

Universal Finite Equ

- $(S_0, T_0) \leq (S, T)$ for some $(S, T) \in R$
- If $\big(P(s), Q(t)\big) \in R$
  with $P(x) := S_1 \cdot S_2$ and $Q(x) := T_1 \cdot T_2$,
  then for $i \in \{1,2\}$,
  $S_i[s/x] = T_i[t/x]$ is a simple term
  or $(S_i[s/x], T_i[t/x]) \leq (S, T)$ for some
  $(S, T) \in R$

- Consider an MGU σ of $P(x)$ and $Q(y)$.
  Then $P(s) \equiv Q(t)$ iff $\big(P(s), Q(t)\big) \leq \mathrm{MGA}(P, Q)$
  where $\mathrm{MGA}(P, Q) := (\sigma P(x), \sigma Q(y))$
- Then $\mathrm{R} := \{\mathrm{MGA}(P, Q) \mid P \text{ and } Q \text{ are unifiable}\}$
  is a finite equivalence proof for all equivalent terms $P(s)$ and $Q(t)$.
- Thus equivalence of terms is semi-decidable.

Decidability of Equivalence

- Equivalence of terms is semi-decidable.
- Non-equivalence is semi-decidable too: the trees must differ at some finite level.
- Thus equivalence is decidable.
- In the next talks, I will present an efficient procedure to decide equivalence by reducing the problem to a fragment of semi-unification.

Literature

Fokkink, W. Unification for infinite sets of equations between finite terms. *Information processing letters 62*, 4 (1997), 183–188.

Sabelfeld, V. The tree equivalence of linear recursion schemes. *Theoretical Computer Science 238*, 1–2 (2000), 1–29.