15-812: Programming Language Semantics

Lecture Notes on Static Semantics

André Platzer

Carnegie Mellon University
Lecture 9

1 Introduction

After we now saw the clever use of uniform substitution as a single mechanism to rigorously justify the correctness of numerous operations in program reasoning, we are now well-motivated to understand the pieces that go into the definition of uniform substitutions. All that they are based on is the static semantics of the free variables and the bound variables of formulas and programs. Classically, the static semantics is studied earlier in the development of programming language semantics [Rey98], but is then potentially rather unmotivated. Now we understand exactly what we need it for and have already seen both useful applications and warning signs of what happens if we get things wrong.

This lecture is based on prior work [Pla15, Pla17]. As usual, we will develop things one step at a time even if this presentation summarizes the result instead of the process. For your long-term understanding, the process is quite impactful to enable you to take advantage of programming language semantics in your favorite application contexts.

2 Dynamic Logic

Recall the syntax and semantics of dynamic logic.

2.1 Syntax

Definition 1 (Terms)**.** Terms are defined by the following grammar (θ, η are terms, x a variable, q a number literal and f a function symbol):

$$\theta, \eta ::= x \mid q \mid f(\theta_1, \ldots, \theta_n) \mid \theta + \eta \mid \theta \cdot \eta$$

Definition 2 (Nondeterministic programs)**.** Programs are defined by the following grammar (α, β are programs, x a variable, θ a term possibly containing x, and Q a first-order formula):

$$\alpha, \beta \ ::= \ x := \theta \mid ?Q \mid \alpha; \beta \mid \mathsf{if}(Q)\, \alpha\, \mathsf{else}\, \beta \mid \mathsf{while}(Q)\, \alpha \mid \alpha^*$$

Definition 3 (DL formula)**.** The *formulas of dynamic logic* (DL) are defined by the grammar (where ϕ, ψ are DL formulas, θ_1, θ_2 terms, p a predicate symbol, x a variable, α a program):

$$\phi, \psi \ ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid p(\theta_1, \ldots, \theta_n) \mid \neg\phi \mid \phi \wedge \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

2.2 Dynamic Semantics

A state is a mapping from variables to $\mathbb{R}$. The set of states is denoted $\mathcal{S}$. Let ω_x^r denote the state that agrees with state ω except for the interpretation of variable x, which is changed to $r \in \mathbb{R}$.

For later use, additional function symbols f and predicate symbols p are already allowed in the logic. For now, consider unary minus $f(x)$ with the fixed interpretation as $I(f)(d) = -d$ and arithmetic comparison operators $p(x, y)$ with the fixed interpretation $(c, d) \in I(p)$ iff $c < d$. But this device of extending DL with additional function symbols and predicate symbols will prove useful later.

Definition 4 (Semantics of terms)**.** The *semantics of a term* θ in a state $\omega \in \mathcal{S}$ is its value. It is defined inductively as follows

- $I\omega[\![x]\!] = \omega(x)$ for variable x
- $I\omega[\![q]\!] = q$ for number literals q
- $I\omega[\![f(\theta_1, \ldots, \theta_k)]\!] = I(f)\big(I\omega[\![\theta_1]\!], \ldots, I\omega[\![\theta_k]\!]\big)$
- $I\omega[\![\theta + \eta]\!] = I\omega[\![\theta]\!] + I\omega[\![\eta]\!]$
- $I\omega[\![\theta \cdot \eta]\!] = I\omega[\![\theta]\!] \cdot I\omega[\![\eta]\!]$

Definition 5 (Transition semantics of programs)**.** Each program α is interpreted semantically as a binary reachability relation $I[\![\alpha]\!] \subseteq \mathcal{S} \times \mathcal{S}$ over states, defined inductively by

- $I[\![x := \theta]\!] = \{(\omega, \nu) \ : \ \nu = \omega$ except that $I\nu[\![x]\!] = I\omega[\![\theta]\!]\}$
- $I[\![?Q]\!] = \{(\omega, \omega) \ : \ \omega \in I[\![Q]\!]$ i.e. $\omega \in I[\![Q]\!]\}$
- $I[\![\alpha; \beta]\!] = I[\![\alpha]\!] \circ I[\![\beta]\!] = \{(\omega, \nu) : (\omega, \mu) \in I[\![\alpha]\!], (\mu, \nu) \in I[\![\beta]\!]\}$
- $I[\![\mathsf{if}(Q)\, \alpha\, \mathsf{else}\, \beta]\!] = I[\![Q]\!] \circ I[\![\alpha]\!] \cup I[\![Q]\!]^{\complement} \circ I[\![\beta]\!] =$
  $\{(\omega, \nu) : (\omega, \nu) \in I[\![\alpha]\!]$ and $\omega \in I[\![Q]\!]\} \cup \{(\omega, \nu) : (\omega, \nu) \in I[\![\beta]\!]$ and $\omega \notin I[\![Q]\!]\}$

- $I[\![\alpha^*]\!] = \big(I[\![\alpha]\!]\big)^* = \bigcup_{n\in\mathbb{N}} I[\![\alpha^n]\!] = \big\{(\omega,\nu) : \text{there are an } n \text{ and states } \mu_0 = \omega, \mu_1, \mu_2, \ldots, \mu_n = \nu \text{ such that } (\mu_i, \mu_{i+1}) \in I[\![\alpha]\!] \text{ for all } 0 \leq i < n\big\}$
  with $\alpha^{n+1} \equiv \alpha^n; \alpha$ and $\alpha^0 \equiv ?true$ and where ρ^* is the reflexive transitive closure of a relation $\rho \subseteq \mathcal{S} \times \mathcal{S}$.

- $I[\![\mathsf{while}(Q)\,\alpha]\!] = \big\{(\omega,\nu) : \text{there are an } n \text{ and states } \mu_0 = \omega, \mu_1, \mu_2, \ldots, \mu_n = \nu$ such that for all $0 \leq i < n$: ① the loop condition is true $\mu_i \in I[\![Q]\!]$ and ② from state μ_i is state μ_{i+1} reachable by running α so $(\mu_i, \mu_{i+1}) \in I[\![\alpha]\!]$ and ③ the loop condition is false $\mu_n \notin I[\![Q]\!]$ in the end$\big\} = \big(I[\![Q]\!] \circ I[\![\alpha]\!]\big)^* \circ I[\![Q]\!]^{\complement}$

Definition 6 (DL semantics)**.** The *semantics of a DL formula* ϕ, for each interpretation I with a corresponding set of states $\mathcal{S}$, is the subset $I[\![\phi]\!] \subseteq \mathcal{S}$ of states in which ϕ is true. It is defined inductively as follows

1. $I[\![p(\theta_1, \ldots, \theta_k)]\!] = \{I, \omega \in \mathcal{S} : (I\omega[\![\theta_1]\!], \ldots, I\omega[\![\theta_k]\!]) \in I(p)\}$
2. $I[\![\theta_1 \geq \theta_2]\!] = \{I, \omega \in \mathcal{S} : I\omega[\![\theta_1]\!] \geq I\omega[\![\theta_2]\!]\}$
3. $I[\![\neg\phi]\!] = (I[\![\phi]\!])^{\complement}$
4. $I[\![\phi \wedge \psi]\!] = I[\![\phi]\!] \cap I[\![\psi]\!]$
5. $I[\![\exists x\, \phi]\!] = \{I, \omega \in \mathcal{S} : \omega^r_x \in I[\![\phi]\!] \text{ for some } r \in \mathbb{R}\}$
6. $I[\![\langle\alpha\rangle\phi]\!] = I[\![\alpha]\!] \circ I[\![\phi]\!] = \{\omega : \nu \in I[\![\phi]\!] \text{ for some } \nu \text{ such that } (\omega,\nu) \in I[\![\alpha]\!]\}$
7. $I[\![[\alpha]\phi]\!] = I[\![\neg\langle\alpha\rangle\neg\phi]\!] = (I[\![\alpha]\!] \circ (I[\![\phi]\!])^{\complement})^{\complement} =$
   $\{\omega : \nu \in I[\![\phi]\!] \text{ for all } \nu \text{ such that } (\omega,\nu) \in I[\![\alpha]\!]\}$

A DL formula ϕ is *valid in* I, written $I \models \phi$, iff $I[\![\phi]\!] = \mathcal{S}$. Formula ϕ is *valid*, $\models \phi$, iff $I \models \phi$ for all interpretations I.

2.3 Static Semantics

The static semantics of DL and programs defines some aspects of their behavior that can be read off directly from their syntactic structure without ever having to run their programs or evaluating their dynamical effects. In this context, the most important aspects of the static semantics concern free or bound occurrences of variables (which are closely related to definitions and uses of variables in compilers). Bound variables are those variables x that are bound by $\forall x$ or $\exists x$, but also those that are bound by modalities such as $[x := 5y]$ or $\langle x := x + 1\rangle$ or $[\mathsf{if}(x \geq 0)\, x := 1\, \mathsf{else}\, x := x + 1]$ or $[\mathsf{if}(x \geq 0)\, x := 1\, \mathsf{else}\, ?true]$.

The notions of free and bound variables are defined by simultaneous induction in the subsequent definitions: free variables for terms (FV(θ)), for formulas (FV(ϕ)), and for programs (FV(α)), as well as bound variables for formulas (BV(ϕ)) and for programs (BV(α)). For programs, there will also be a need to distinguish variables that are bound/written to on all executions of α, these are the must-bound variables (MBV(α)).

Must-bound variables are in contrast to the (may) bound variables ($\mathrm{BV}(\alpha)$) which may only be bound on some execution paths of α, such as in $[\mathsf{if}(x \geq 0)\, x := 1\, \mathsf{else}\, (x := 0; y := x + 1)]$, which has bound variables $\{x, y\}$ but must-bound variables only $\{x\}$, because y is not written to in the first choice.

Definition 7 (Bound variable)**.** The set $\mathrm{BV}(\phi)$ of *bound variables* of DL formula ϕ is defined inductively as

$$\begin{aligned}
\mathrm{BV}(\theta_1 \geq \theta_2) &= \emptyset \\
\mathrm{BV}(p(\theta_1, \ldots, \theta_n)) &= \emptyset \\
\mathrm{BV}(\neg\phi) &= \mathrm{BV}(\phi) \\
\mathrm{BV}(\phi \wedge \psi) &= \mathrm{BV}(\phi) \cup \mathrm{BV}(\psi) \\
\mathrm{BV}(\forall x\, \phi) &= \{x\} \cup \mathrm{BV}(\phi) \\
\mathrm{BV}(\exists x\, \phi) &= \{x\} \cup \mathrm{BV}(\phi) \\
\mathrm{BV}([\alpha]\phi) &= \mathrm{BV}(\alpha) \cup \mathrm{BV}(\phi) \\
\mathrm{BV}(\langle\alpha\rangle\phi) &= \mathrm{BV}(\alpha) \cup \mathrm{BV}(\phi)
\end{aligned}$$

Definition 8 (Free variable)**.** The set $\mathrm{FV}(\theta)$ of *free variables* of term θ, i.e. those that occur in θ, is defined inductively as

$$\begin{aligned}
\mathrm{FV}(x) &= \{x\} \\
\mathrm{FV}(q) &= \emptyset \\
\mathrm{FV}(f(\theta_1, \ldots, \theta_n)) &= \mathrm{FV}(\theta_1) \cup \cdots \cup \mathrm{FV}(\theta_n) \\
\mathrm{FV}(\theta + \eta) &= \mathrm{FV}(\theta) \cup \mathrm{FV}(\eta) \\
\mathrm{FV}(\theta \cdot \eta) &= \mathrm{FV}(\theta) \cup \mathrm{FV}(\eta)
\end{aligned}$$

The set $\mathrm{FV}(\phi)$ of *free variables* of DL formula ϕ, i.e. all those that occur in ϕ outside the scope of quantifiers or modalities binding it, is defined inductively as

$$\begin{aligned}
\mathrm{FV}(\theta_1 \geq \theta_2) &= \mathrm{FV}(\theta_1) \cup \mathrm{FV}(\theta_2) \\
\mathrm{FV}(p(\theta_1, \ldots, \theta_n)) &= \mathrm{FV}(\theta_1) \cup \cdots \cup \mathrm{FV}(\theta_n) \\
\mathrm{FV}(\neg\phi) &= \mathrm{FV}(\phi) \\
\mathrm{FV}(\phi \wedge \psi) &= \mathrm{FV}(\phi) \cup \mathrm{FV}(\psi) \\
\mathrm{FV}(\forall x\, \phi) &= \mathrm{FV}(\phi) \setminus \{x\} \\
\mathrm{FV}(\exists x\, \phi) &= \mathrm{FV}(\phi) \setminus \{x\} \\
\mathrm{FV}([\alpha]\phi) &= \mathrm{FV}(\alpha) \cup (\mathrm{FV}(\phi) \setminus \mathrm{MBV}(\alpha)) \\
\mathrm{FV}(\langle\alpha\rangle\phi) &= \mathrm{FV}(\alpha) \cup (\mathrm{FV}(\phi) \setminus \mathrm{MBV}(\alpha))
\end{aligned}$$

The simpler definition $\mathrm{FV}([\alpha]\phi) = \mathrm{FV}(\alpha) \cup \mathrm{FV}(\phi)$ would be correct, but the results are less precise then. Likewise for $\langle\alpha\rangle\phi$.

Which variables are free so may possibly be read ($\mathrm{FV}(\alpha)$), which variables are bound ($\mathrm{BV}(\alpha)$) so may be written to somewhere in α and which variables are must-bound ($\mathrm{MBV}(\alpha)$) so must be written to on all execution paths.

Definition 9 (Bound variable)**.** The set $\mathrm{BV}(\alpha)$ of *bound variables* of program α, i.e. all those that may potentially be written to, is defined inductively as

$$\begin{aligned}\mathrm{BV}(x := \theta) &= \{x\}\\ \mathrm{BV}(?Q) &= \emptyset\\ \mathrm{BV}(\alpha;\beta) &= \mathrm{BV}(\alpha) \cup \mathrm{BV}(\beta)\\ \mathrm{BV}(\mathsf{if}(Q)\,\alpha\,\mathsf{else}\,\beta) &= \mathrm{BV}(\alpha) \cup \mathrm{BV}(\beta)\\ \mathrm{BV}(\alpha^*) &= \mathrm{BV}(\alpha)\end{aligned}$$

Definition 10 (Must-bound variable)**.** The set $\mathrm{MBV}(\alpha)$ of *must-bound variables* of program α, i.e. all those that must be written to on all paths of α, is defined inductively as

$$\begin{aligned}\mathrm{MBV}(x := \theta) &= \{x\}\\ \mathrm{MBV}(?Q) &= \emptyset\\ \mathrm{MBV}(\alpha;\beta) &= \mathrm{MBV}(\alpha) \cup \mathrm{MBV}(\beta)\\ \mathrm{MBV}(\mathsf{if}(Q)\,\alpha\,\mathsf{else}\,\beta) &= \mathrm{MBV}(\alpha) \cap \mathrm{MBV}(\beta)\\ \mathrm{MBV}(\alpha^*) &= \emptyset\end{aligned}$$

Obviously, $\mathrm{MBV}(\alpha) \subseteq \mathrm{BV}(\alpha)$. If α is only built by sequential compositions from atomic programs, then $\mathrm{MBV}(\alpha) = \mathrm{BV}(\alpha)$.

Definition 11 (Free variable)**.** The set $\mathrm{FV}(\alpha)$ of *free variables* of program α, i.e. all those that may potentially be read, is defined inductively as

$$\begin{aligned}\mathrm{FV}(x := \theta) &= \mathrm{FV}(\theta)\\ \mathrm{FV}(?Q) &= \mathrm{FV}(Q)\\ \mathrm{FV}(\alpha;\beta) &= \mathrm{FV}(\alpha) \cup (\mathrm{FV}(\beta) \setminus \mathrm{MBV}(\alpha))\\ \mathrm{FV}(\mathsf{if}(Q)\,\alpha\,\mathsf{else}\,\beta) &= \mathrm{FV}(Q) \cup \mathrm{FV}(\alpha) \cup \mathrm{FV}(\beta)\\ \mathrm{FV}(\alpha^*) &= \mathrm{FV}(\alpha)\end{aligned}$$

The *variables* of program α are $\mathrm{V}(\alpha) = \mathrm{FV}(\alpha) \cup \mathrm{BV}(\alpha)$.

The simpler definition $\mathrm{FV}(\alpha;\beta) = \mathrm{FV}(\alpha) \cup \mathrm{FV}(\beta)$ would be correct, but the results are less precise then.

Note that these definitions are sound syntactic overapproximations of the concept of reading and writing variables. There may not be any actual run of α in which the variable is read, nor one in which is actually written to. But without the program α mentioning x as a free variable $x \in \mathrm{FV}(\alpha)$, it cannot read the value of x. Likewise, without α mentioning x as a bound variable $x \in \mathrm{BV}(\alpha)$, it cannot write to x. The definition Def. 11 is already parsimonious. For example, x is not a free variable of the following program

$$(\mathsf{if}(y \geq 0)\, x := 1\, \mathsf{else}\, x := 2); z := x + y$$

because x is never actually read, since x must have been defined on *every* execution path of the first part before being read by the second part. No execution of the above program, thus, depends on the initial value of x, which is why it is not a free variable. This would have been different for the simpler definition

$$\text{FV}(\alpha;\beta) = \text{FV}(\alpha) \cup \text{FV}(\beta)$$

There is a limit to the precision with which any such static analysis of the program can determine which variables are really read and really written, though, by Rice's theorem [Ric53]. The static semantics in Def. 11 will, e.g., call x a free variable of the following program even though no execution could ever read it, because they all fail the test $?\mathit{false}$ when trying to run the branch that reads x:

$$z := 0; (?\mathit{false}; z := z + x)^*$$

The following insight reflects that for a variable x to be modified during a run of α, the program α first needs to have x be a bound variable, i.e. $x \in \text{BV}(\alpha)$, so that α can write to x. The converse is not true, because α may be binding a variable x, e.g. by having an assignment to x, that it never actually changes, because that assignment is never executed. The following program, for example, binds x but will never actually change the value of x because there is no way of satisfying the test $?\mathit{false}$:

$$\mathsf{if}(y \geq 0)\ (?\mathit{false}; x := 42)\ \mathsf{else}\ z := x + 1$$

Lemma 12 (Bound lemma)**.** *If $(\omega, \nu) \in I[\![\alpha]\!]$, then $\omega = \nu$ on $BV(\alpha)^{\complement}$.*

Proof. The proof is by a straightforward structural induction on α.

- $(\omega, \nu) \in I[\![x := \theta]\!] = \{(\omega, \nu) : \nu = \omega \text{ except that } I\nu[\![x]\!] = I\omega[\![\theta]\!]\}$ implies that $\omega = \nu$ except for $\{x\} = \text{BV}(x := \theta)$.
- $(\omega, \omega) \in I[\![?Q]\!] = \{(\omega, \omega) : \omega \in I[\![Q]\!] \text{ i.e. } \omega \in I[\![Q]\!]\}$ fits to $\text{BV}(?Q) = \emptyset$
- $(\omega, \nu) \in I[\![\alpha;\beta]\!] = I[\![\alpha]\!] \circ I[\![\beta]\!]$, i.e. there is a μ such that $(\omega, \mu) \in I[\![\alpha]\!]$ and $(\mu, \nu) \in I[\![\beta]\!]$. So, by induction hypothesis, $\omega = \mu$ on $\text{BV}(\alpha)^{\complement}$ and $\mu = \nu$ on $\text{BV}(\beta)^{\complement}$. By transitivity, $\omega = \nu$ on $(\text{BV}(\alpha) \cup \text{BV}(\beta))^{\complement} = \text{BV}(\alpha;\beta)^{\complement}$.
- $(\omega, \nu) \in I[\![\mathsf{if}(Q)\ \alpha\ \mathsf{else}\ \beta]\!]$ is left as an exercise.
- $(\omega, \nu) \in I[\![\alpha^*]\!] = \bigcup_{n \in \mathbb{N}} I[\![\alpha^n]\!]$, then there is an $n \in \mathbb{N}$ and a sequence $\nu_0 = \omega, \nu_1, \ldots, \nu_n = \nu$ such that $(\nu_i, \nu_{i+1}) \in I[\![\alpha]\!]$ for all $i < n$. By n uses of the induction hypothesis, $\nu_i = \nu_{i+1}$ on $\text{BV}(\alpha)^{\complement}$ for all $i < n$. Thus, $\omega = \nu_0 = \nu_n = \nu$ on $\text{BV}(\alpha)^{\complement} = \text{BV}(\alpha^*)^{\complement}$.

□

The value of terms only depend on the values of their free variables. When evaluating a term θ in two states ω, $\tilde{\nu}$ that differ widely but agree on the free variables $\text{FV}(\theta)$ of θ, then the values of θ in both states coincide.

Lemma 13 (Coincidence lemma). *If $\omega = \tilde{\nu}$ on $FV(\theta)$, then $I\omega[\![\theta]\!] = I\tilde{\nu}[\![\theta]\!]$.*

Proof. The proof is by a straightforward structural induction on θ.

- $I\omega[\![x]\!] = \omega(x) = \tilde{\nu}(x) = I\tilde{\nu}[\![x]\!]$ for variable x since $\omega = \tilde{\nu}$ on $\mathrm{FV}(x) = \{x\}$.
- $I\omega[\![q]\!] = q = I\tilde{\nu}[\![q]\!]$ for number literals q.
- $I\omega[\![\theta + \eta]\!] = I\omega[\![\theta]\!] + I\omega[\![\eta]\!] = I\tilde{\nu}[\![\theta]\!] + I\tilde{\nu}[\![\eta]\!] = I\tilde{\nu}[\![\theta + \eta]\!]$ by induction hypothesis, because $\mathrm{FV}(\theta) \subseteq \mathrm{FV}(\theta + \eta)$ and $\mathrm{FV}(\eta) \subseteq \mathrm{FV}(\theta + \eta)$.
- $I\omega[\![\theta \cdot \eta]\!] = I\omega[\![\theta]\!] \cdot I\omega[\![\eta]\!] = I\tilde{\nu}[\![\theta]\!] \cdot I\tilde{\nu}[\![\eta]\!] = I\tilde{\nu}[\![\theta \cdot \eta]\!]$ by induction hypothesis, because $\mathrm{FV}(\theta) \subseteq \mathrm{FV}(\theta \cdot \eta)$ and $\mathrm{FV}(\eta) \subseteq \mathrm{FV}(\theta \cdot \eta)$.

□

By a more subtle argument, the value of DL formulas also only depend on the values of their free variables. When evaluating DL formula ϕ in two states ω, $\tilde{\nu}$ that differ widely but agree on the free variables $\mathrm{FV}(\phi)$ of ϕ, then the (truth) values of ϕ in both states coincide. Lemma 14 and 16 are proved by simultaneous induction.

Lemma 14 (Coincidence lemma). *If $\omega = \tilde{\nu}$ on $FV(\phi)$, then $\omega \in I[\![\phi]\!]$ iff $\tilde{\nu} \in I[\![\phi]\!]$.*

Proof. The proof is by induction on the structural complexity of ϕ.

1. $\omega \in I[\![p(\theta_1, \ldots, \theta_k)]\!]$ iff $(I\omega[\![\theta_1]\!], \ldots, I\omega[\![\theta_k]\!]) \in I(p)$ iff $(I\tilde{\nu}[\![\theta_1]\!], \ldots, I\tilde{\nu}[\![\theta_k]\!]) \in I(p)$ iff $\tilde{\nu} \in I[\![p(\theta_1, \ldots, \theta_k)]\!]$ by Lemma 13 since $\mathrm{FV}(\theta_i) \subseteq \mathrm{FV}(p(\theta_1, \ldots, \theta_k))$.
2. $\omega \in I[\![\theta_1 \geq \theta_2]\!]$ iff $I\omega[\![\theta_1]\!] \geq I\omega[\![\theta_2]\!]$ iff $I\tilde{\nu}[\![\theta_1]\!] \geq I\tilde{\nu}[\![\theta_2]\!]$ iff $\tilde{\nu} \in I[\![\theta_1 \geq \theta_2]\!]$ by Lemma 13 since $\mathrm{FV}(\theta_i) \subseteq \mathrm{FV}(\theta_1 \geq \theta_2)$.
3. $\omega \in I[\![\neg\phi]\!]$ iff $\omega \notin I[\![\phi]\!]$ iff $\tilde{\nu} \notin I[\![\phi]\!]$ iff $\tilde{\nu} \in I[\![\neg\phi]\!]$ by induction hypothesis as $\mathrm{FV}(\neg\phi) = \mathrm{FV}(\phi)$.
4. $\omega \in I[\![\phi \wedge \psi]\!]$ iff $\omega \in I[\![\phi]\!] \cap I[\![\psi]\!]$ iff $\tilde{\nu} \in I[\![\phi]\!] \cap I[\![\psi]\!]$ iff $\tilde{\nu} \in I[\![\phi \wedge \psi]\!]$ by induction hypothesis using $\mathrm{FV}(\phi \wedge \psi) = \mathrm{FV}(\phi) \cup \mathrm{FV}(\psi)$.
5. $\omega \in I[\![\exists x\, \phi]\!]$ iff $\omega^r_x \in I[\![\phi]\!]$ for some $r \in \mathbb{R}$ iff $\tilde{\nu}^r_x \in I[\![\phi]\!]$ for some $r \in \mathbb{R}$ iff $\tilde{\nu} \in I[\![\exists x\, \phi]\!]$ by induction hypothesis using that, for the same r, $\omega^r_x = \tilde{\nu}^r_x$ on $\mathrm{FV}(\phi) \subseteq \{x\} \cup \mathrm{FV}(\exists x\, \phi)$.
6. $\omega \in I[\![\langle\alpha\rangle\phi]\!]$ will be postponed until Sect. **??**
7. $\omega \in I[\![[\alpha]\phi]\!] = I[\![\neg\langle\alpha\rangle\neg\phi]\!]$ iff $\omega \notin I[\![\langle\alpha\rangle\neg\phi]\!]$ iff $\tilde{\nu} \notin I[\![\langle\alpha\rangle\neg\phi]\!]$ iff $\tilde{\nu} \in I[\![[\alpha]\phi]\!]$ by induction hypothesis using $\mathrm{FV}(\langle\alpha\rangle\neg\phi) = \mathrm{FV}([\alpha]\phi)$.

□

In a sense, the runs of an program α also only depend on the values of its free variables, because its behavior cannot depend on the values of variables that it never reads. That is, if $\omega = \tilde{\nu}$ on $\mathrm{FV}(\alpha)$ and $(\omega, \nu) \in I[\![\alpha]\!]$, then there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in I[\![\alpha]\!]$ and ν and $\tilde{\omega}$ agree in some sense. There is a subtlety, though. The resulting states ν and $\tilde{\omega}$ will only continue to agree on $\mathrm{FV}(\alpha)$ and the variables that are bound on the particular path that α took for the transition $(\omega, \nu) \in I[\![\alpha]\!]$. On variables z that are neither free (so the initial states ω and $\tilde{\nu}$ have not been assumed to already coincide on z) nor bound on the particular path that α took, ν and $\tilde{\omega}$ may continue to disagree, because z has not been written to on that path.

Example 15. Let $(\omega, \nu) \in I[\![\alpha]\!]$. It is not enough to assume $\omega = \tilde{\nu}$ only on $\mathrm{FV}(\alpha)$ in order to guarantee $\nu = \tilde{\omega}$ on $\mathrm{V}(\alpha)$ for some $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in I[\![\alpha]\!]$, because

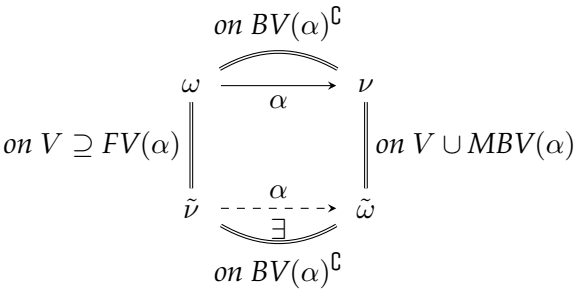
$$\alpha \overset{\text{def}}{\equiv} (z := z + 1)^*; \mathsf{if}(z > 0)\, x := 1 \,\mathsf{else}\, y := 2$$

will force the final states to agree only on either x or on y, whichever one was assigned to during the respective run of α, not on all $\mathrm{BV}(\alpha) = \{x, y, z\}$, even though any initial states $\omega, \tilde{\nu}$ agree on $\mathrm{FV}(\alpha) = \emptyset$. Note that this can only happen because $\mathrm{MBV}(\alpha) = \{z\} \neq \mathrm{BV}(\alpha) = \{x, y, z\}$.

Yet, ν and $\tilde{\omega}$ will certainly agree on the variables that are bound on *all* paths of α, rather than just somewhere in α. These are the must-bound variables of α.

If initial states agree on (at least) all free variables $\mathrm{FV}(\alpha)$ that program α may read, then the final states agree on those as well as on all variables that α must write, i.e. on $\mathrm{MBV}(\alpha)$.

Lemma 16 (Coincidence lemma)**.** *If $\omega = \tilde{\nu}$ on $V \supseteq FV(\alpha)$ and $(\omega, \nu) \in I[\![\alpha]\!]$, then there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in I[\![\alpha]\!]$ and $\nu = \tilde{\omega}$ on $V \cup MBV(\alpha)$.*



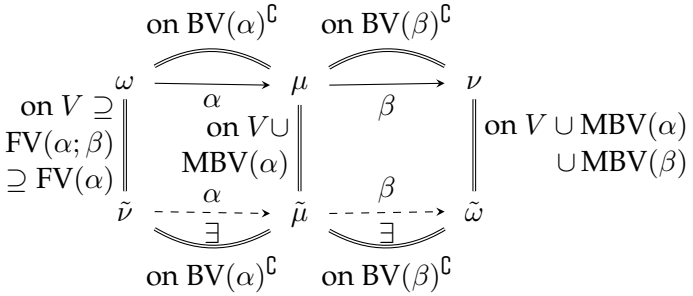
Proof. The proof is by induction on the structural complexity of α, where α^* is considered to be structurally more complex than programs of any length but with less repetitions, which induces a well-founded order on programs. For atomic programs α, for which $\mathrm{BV}(\alpha) = \mathrm{MBV}(\alpha)$, it is enough to conclude agreement on $\mathrm{V}(\alpha) = \mathrm{FV}(\alpha) \cup \mathrm{BV}(\alpha) = \mathrm{FV}(\alpha) \cup \mathrm{MBV}(\alpha)$, because any variable in $V \setminus \mathrm{V}(\alpha)$ is in $\mathrm{BV}(\alpha)^\complement$, which remains unchanged by α according to Lemma 12.

- $(\omega, \nu) \in I[\![x := \theta]\!] = \{(\omega, \nu) : \nu = \omega$ except that $I\nu[\![x]\!] = I\omega[\![\theta]\!]\}$ then there is a transition $(\tilde{\nu}, \tilde{\omega}) \in I[\![x := \theta]\!]$ and $\tilde{\omega}(x) = I\tilde{\omega}[\![x]\!] = I\tilde{\nu}[\![\theta]\!] = I\omega[\![\theta]\!] = I\nu[\![x]\!] = \omega(x)$ by

Lemma 14, since $\omega = \tilde{\nu}$ on $\mathrm{FV}(x := \theta) = \mathrm{FV}(\theta)$. So, $\nu = \tilde{\omega}$ on $\mathrm{BV}(x := \theta) = \{x\}$. Also, $\omega = \nu$ on $\mathrm{BV}(x := \theta)^\complement$ and $\tilde{\nu} = \tilde{\omega}$ on $\mathrm{BV}(x := \theta)^\complement$ by Lemma 12. Since $\omega = \tilde{\nu}$ on $\mathrm{FV}(x := \theta)$, these imply $\nu = \tilde{\omega}$ on $\mathrm{FV}(x := \theta) \setminus \mathrm{BV}(x := \theta)$. Since $\nu = \tilde{\omega}$ on $\mathrm{BV}(x := \theta)$ had been shown already, this implies $\nu = \tilde{\omega}$ on $\mathrm{V}(x := \theta)$.

- $(\omega, \nu) \in I[\![?Q]\!] = \{(\omega, \omega) : \omega \in I[\![Q]\!] \text{ i.e. } \omega \in I[\![Q]\!]\}$ then $\nu = \omega$ by Def. 5. Since, $\omega \in I[\![Q]\!]$ and $\omega = \tilde{\nu}$ on $\mathrm{FV}(?Q)$, Lemma 14 implies that $\tilde{\nu} \in I[\![Q]\!]$, so $(\tilde{\nu}, \tilde{\nu}) \in I[\![?Q]\!]$. Finally, $\omega = \tilde{\nu}$ on $\mathrm{V}(?Q)$ follows since $\mathrm{BV}(?Q) = \emptyset$.

- $(\omega, \nu) \in I[\![\alpha; \beta]\!] = I[\![\alpha]\!] \circ I[\![\beta]\!]$, i.e. there is a μ such that $(\omega, \mu) \in I[\![\alpha]\!]$ and $(\mu, \nu) \in I[\![\beta]\!]$. Since $V \supseteq \mathrm{FV}(\alpha; \beta) \supseteq \mathrm{FV}(\alpha)$, by induction hypothesis, there is a $\tilde{\mu}$ such that $(\tilde{\nu}, \tilde{\mu}) \in I[\![\alpha]\!]$ and $\mu = \tilde{\mu}$ on $V \cup \mathrm{MBV}(\alpha)$. Since $V \supseteq \mathrm{FV}(\alpha; \beta)$, so $V \cup \mathrm{MBV}(\alpha) \supseteq \mathrm{FV}(\alpha; \beta) \cup \mathrm{MBV}(\alpha) = \mathrm{FV}(\alpha) \cup (\mathrm{FV}(\beta) \setminus \mathrm{MBV}(\alpha)) \cup \mathrm{MBV}(\alpha) = \mathrm{FV}(\alpha) \cup \mathrm{FV}(\beta) \cup \mathrm{MBV}(\alpha) \supseteq \mathrm{FV}(\beta)$ by Def. 11, and since $(\mu, \nu) \in I[\![\beta]\!]$, the induction hypothesis implies that there is an $\tilde{\omega}$ such that $(\tilde{\mu}, \tilde{\omega}) \in I[\![\beta]\!]$ and $\nu = \tilde{\omega}$ on $(V \cup \mathrm{MBV}(\alpha)) \cup \mathrm{MBV}(\beta) = V \cup \mathrm{MBV}(\alpha; \beta)$.



- $(\omega, \nu) \in I[\![\mathsf{if}(Q)\, \alpha\, \mathsf{else}\, \beta]\!]$ is left as an exercise.

- $(\omega, \nu) \in I[\![\alpha^*]\!] = \bigcup_{n \in \mathbb{N}} I[\![\alpha^n]\!]$ iff there is an $n \in \mathbb{N}$ such that $(\omega, \nu) \in I[\![\alpha^n]\!]$. The case $n = 0$ follows from the assumption $\omega = \tilde{\nu}$ on $V \supseteq \mathrm{FV}(\alpha)$, since $\nu = \omega$ holds in that case and $\mathrm{MBV}(\alpha^*) = \emptyset$. The case $n > 0$ proceeds as follows. Since $\mathrm{FV}(\alpha^n) = \mathrm{FV}(\alpha^*) = \mathrm{FV}(\alpha)$, the induction hypothesis applied to the structurally simpler program α^n implies that there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in I[\![\alpha^n]\!]$ and $\nu = \tilde{\omega}$ on $V \cup \mathrm{MBV}(\alpha^n) \supseteq V = V \cup \mathrm{MBV}(\alpha^*)$, since $\mathrm{MBV}(\alpha^*) = \emptyset$. Since $I[\![\alpha^n]\!] \subseteq I[\![\alpha^*]\!]$, this concludes the proof.

□

When assuming ω and $\tilde{\nu}$ to agree on all $\mathrm{V}(\alpha)$, whether free or bound, ν and $\tilde{\omega}$ will continue to agree on $\mathrm{V}(\alpha)$:

Corollary 17 (Coincidence lemma)**.** *If $\omega = \tilde{\nu}$ on $V(\alpha)$ and $(\omega, \nu) \in I[\![\alpha]\!]$, then there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in I[\![\alpha]\!]$ and $\nu = \tilde{\omega}$ on $V(\alpha)$. The same continues to hold if $\omega = \tilde{\nu}$ only on $V(\alpha) \setminus MBV(\alpha)$.*

Proof. By Lemma 16 using either $V \overset{\text{def}}{=} \mathrm{V}(\alpha) \supseteq \mathrm{FV}(\alpha)$ or $V \overset{\text{def}}{=} \mathrm{V}(\alpha) \setminus \mathrm{MBV}(\alpha)$, respectively. □

References

[Pla15] André Platzer. A uniform substitution calculus for differential dynamic logic. In Amy Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 467–481, Berlin, 2015. Springer. doi:10.1007/978-3-319-21401-6_32.

[Pla17] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.*, 59(2):219–265, 2017. doi:10.1007/s10817-016-9385-1.

[Rey98] John C. Reynolds. *Theories of Programming Languages*. Cambridge Univ. Press, 1998.

[Ric53] H. Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Trans. AMS*, 74(2):358–366, 1953. doi:10.2307/1990888.