Dynamic Logic

Part 1: Programs and Their Semantics

Wolfgang Poiger and Igor Sedlár

Institute of Computer Science
Czech Academy of Sciences

Faculty of Arts, Charles University
Fall Semester 2025-26

Course overview – 1

The course introduces some logics for reasoning about the properties of computer programs (mostly equivalence and correctness). Structure:

1. Program semantics and Hoare Logic
2. Propositional Dynamic Logic
3. Kleene algebra

Instructors:

- Igor Sedlár (Parts 1 and 3) `sedlar@cs.cas.cz`
- Wolfgang Poiger (Part 2) `poiger@cs.cas.cz`

Both at the Institute of Computer Science, Czech Academy of Sciences
(Pod Vodárenskou věží 2, Ládví).

Course overview – 2

The course will be taught in English.

The fail/pass decision will be based on

- solution of 3 problem sets (roughly: late Oct, late Nov, mid-Jan)
- lecture attendance

Course materials etc. at the course webpage:

Equivalence of programs: A motivating example

(i)

```
def print_primes(y):
  x := 2
  while x ≤ y do
    if is_prime(x) then
      print(x)
      x := x + 1
    else
      x := x + 1
    end if
  end while
```

(ii)

```
def print_primes(y):
  x := 2
  while x ≤ y do
    if is_prime(x) then
      print(x)
    end if
    x := x + 1
  end while
```

Figure: Two programs for printing out primes.

A formal language of programs

Variables: propositions $\Pi = \{\mathtt{p}_1, \mathtt{p}_2, \ldots\}$, programs/actions $\Sigma = \{\mathtt{a}_1, \mathtt{a}_2, \ldots\}$.

Definition 1

Boolean formulas (Fm)		*Program expressions* (Pr)	
$B, C ::= \mathtt{p} \in \Pi$		$E, F ::= \mathtt{a} \in \Sigma$	
$\mid \top$	*true*	$\mid \mathbf{skip}$	*"Do nothing" / "Wait"*
$\mid \bot$	*false*	$\mid \mathbf{abort}$	*"Stop the computation"*
$\mid \neg B$	*not*	$\mid E; F$	*"Do E, then do F"*
$\mid B \wedge C$	*and*	$\mid \mathbf{if}\ B\ \mathbf{then}\ E\ \mathbf{else}\ F$	*Conditionals*
$\mid B \vee C$	*or*	$\mid \mathbf{while}\ B\ \mathbf{do}\ E$	*While loops*

Example 1: (i) formalised

$\mathtt{a}; (\mathbf{while}\ \mathtt{p}\ \mathbf{do}\ (\mathbf{if}\ \mathtt{q}\ \mathbf{then}\ \mathtt{b}; \mathtt{c}\ \mathbf{else}\ \mathtt{c}))$

where $\mathtt{a} \leftarrow (x := 2)$, $\mathtt{b} \leftarrow \mathtt{print}(x)$, $\mathtt{c} \leftarrow (x := x + 1)$ and $\mathtt{p} \leftarrow (x \leq y)$, $\mathtt{q} \leftarrow \mathtt{is\_prime}(x)$

Relational semantics – 1

Definition 2

A relational model for programs is $M = \langle X, \mathsf{sat}_M, \mathsf{rel}_M \rangle$ *where*

- $X \neq \emptyset$
- $\mathsf{sat}_M : \Pi \to \mathcal{P}(X)$
- $\mathsf{rel}_M : \Sigma \to \mathcal{P}(X \times X)$

sat_M *generalizes to* $Fm \to \mathcal{P}(X)$ *in the usual way.*

Intuition: X is a set of "states"; $\mathsf{sat}_M(\mathtt{p})$ is the set of states where $\mathtt{p}$ "is satisfied" and $\mathsf{rel}_M(\mathtt{a})$ is the "input-output relation" for $\mathtt{a}$. That is, $\langle x, y \rangle \in \mathsf{rel}_M(\mathtt{a})$ iff $\mathtt{a}$ may halt in state y when executed in x.

Note: $\mathsf{rel}_M(\mathtt{a})$ is not necessarily a (total) function (non-determinism).

Relational semantics – 2

Example 2

A relational model for the motivating example:

- $X = \mathbb{N} \times \mathbb{N} \times \overbrace{\mathbb{N}^*}^{\text{finite sequences over } \mathbb{N}}$
- $\mathsf{sat}(\mathtt{p}) = \{\langle n, m, s\rangle \mid n \leq m\}$
- $\mathsf{sat}(\mathtt{q}) = \{\langle n, m, s\rangle \mid n \text{ is prime}\}$
- and rel is the minimal relation such that

$$\langle n, m, s\rangle \xrightarrow{\mathtt{a}} \langle 2, m, s\rangle \qquad \langle n, m, s\rangle \xrightarrow{\mathtt{b}} \langle n, m, sn\rangle$$

$$\langle n, m, s\rangle \xrightarrow{\mathtt{c}} \langle n+1, m, s\rangle$$

Recall: $\mathtt{a} \leftarrow (x := 2)$, $\mathtt{b} \leftarrow \mathtt{print}(x)$, $\mathtt{c} \leftarrow (x := x + 1)$ and $\mathtt{p} \leftarrow (x \leq y)$, $\mathtt{q} \leftarrow \mathtt{is\_prime}(x)$

Relational semantics – 3

Recall: If $R, R_1, R_2 \subseteq X \times X$, then

- $R_1 \circ R_2 = \{\langle x, y\rangle \mid \exists z \in X : \langle x, z\rangle \in R_1 \ \ \& \ \ \langle z, y\rangle \in R_2\}$
- $R^* = \bigcup_{n \geq 0} R^n$, where $R^0 = 1_X = \{\langle x, x\rangle \mid x \in X\}$ and $R^{n+1} = R^n \circ R$.

Definition 3

Given M, we define $[\![-]\!]_M : Fm \cup Pr \to \mathcal{P}(X \times X)$:

- $[\![B]\!]_M = 1_{\mathsf{sat}_M(B)}$ *and* $[\![\mathtt{a}]\!]_M = \mathsf{rel}_M(\mathtt{a})$ *for* $B \in Fm$, $\mathtt{a} \in \Sigma$
- $[\![\mathbf{skip}]\!]_M = 1_X$ *and* $[\![\mathbf{abort}]\!]_M = \emptyset$
- $[\![E; F]\!]_M = [\![E]\!]_M \circ [\![F]\!]_M$
- $[\![\mathbf{if}\ B\ \mathbf{then}\ E\ \mathbf{else}\ F]\!]_M = ([\![B]\!]_M \circ [\![E]\!]_M) \cup ([\![\neg B]\!]_M \circ [\![F]\!]_M)$
- $[\![\mathbf{while}\ B\ \mathbf{do}\ E]\!]_M = ([\![B]\!]_M \circ [\![E]\!]_M)^* \circ [\![\neg B]\!]_M$

Programs P and Q are relationally equivalent iff $[\![P]\!]_M = [\![Q]\!]_M$ for all M.

(Notation: $P \equiv Q$).

Relational semantics – 4

Example 3

Complex programs in Example 2:

- $\langle n, m, s\rangle \xrightarrow{\texttt{b ; c}} \langle n+1, m, ns\rangle$
- $\langle 2, 3, s\rangle \xrightarrow{\textbf{if p then b else c}} \langle 2, 3, s2\rangle \quad \langle 3, 2, s\rangle \xrightarrow{\textbf{if p then b else c}} \langle 4, 2, s\rangle$
- $\langle 2, 4, \epsilon\rangle \xrightarrow{\textbf{while p do b;c}} \langle 5, 4, 2\,3\,4\rangle \quad \langle 4, 2, \epsilon\rangle \xrightarrow{\textbf{while p do b;c}} \langle 4, 2, \epsilon\rangle$

Relational semantics – 5

Simplifying notation:

$$\mathbf{skip} = 1 \qquad \mathbf{abort} = 0$$

$$\mathbf{if}\ B\ \mathbf{then}\ E\ \mathbf{else}\ F = E +_B F \qquad \mathbf{while}\ B\ \mathbf{do}\ E = E^{(B)}$$

We define $\mathbf{assert}\ B := 1 +_B 0$. We usually write "$B$" instead of "$\mathbf{assert}\ B$".

We define $\mathcal{H} : Pr \to Fm$ (the halt predicate):

$$\mathcal{H}(\mathsf{a}) := \bot \quad \mathcal{H}(\mathbf{skip}) := \top \quad \mathcal{H}(\mathbf{abort}) := \bot \quad \mathcal{H}(E;F) := \mathcal{H}(E) \wedge \mathcal{H}(F)$$

$$\mathcal{H}(E +_B F) := (B \wedge \mathcal{H}(E)) \vee (\neg B \wedge \mathcal{H}(F)) \qquad \mathcal{H}(E^{(B)}) := \neg B.$$

Hence: $\mathcal{H}(\mathbf{assert}\ B) = (B \wedge \top) \vee (\neg B \wedge \bot) \equiv B$.

Relational semantics – 6

Proposition 1

GKAT axioms are relationally valid:

BA. $B \equiv C$ *valid in* BA

U1. $E +_B E \equiv E$

U2. $E +_B F \equiv F +_{(\neg B)} E$

U3. $(E +_B F) +_C G \equiv E +_{(B \wedge C)} (F +_C G)$

U4. $E +_B F \equiv B; E +_B F$

U5. $(E +_B F); G \equiv (E; G) +_B (F; G)$

S1. $E; (F; G) \equiv (E; F); G$

S2. $0; E \equiv 0$

S3. $E; 0 \equiv 0$

S4. $1; E \equiv E$

S5. $E; 1 \equiv E$

W1. $E^{(B)} \equiv E; E^{(B)} +_B 1$

W2. $(E +_C 1)^{(B)} \equiv (C; E)^{(B)}$

W3. $\dfrac{G \equiv E; G +_B F}{G \equiv E^{(B)}; F}$ *if* $\mathcal{H}(E) = \bot$

Example 4

Hence, $((E; F) +_C F)^{(B)} \equiv ((E +_C 1); F)^{(B)}$.

Trace semantics – 1

A state is a complete and consistent set of literals (containing exactly one of $\mathtt{p}$ and $\overline{\mathtt{p}}$ for each $\mathtt{p} \in \Pi$). We write $S \vDash \mathtt{r}$ if $\mathtt{r} \in S$. ($S \vDash B$ as expected.)

Definition 4

A trace (over Π and Σ) is a sequence of the form

$$S_1 \mathtt{a}_1 S_2 \ldots \mathtt{a}_{n-1} S_n$$

where $n \geq 1$, each S_i is a state (over Π) and each $\mathtt{a}_j \in \Sigma$. Tr is the set of all traces.

Note: If Π is finite, then each state is a word over the set of literals and each trace is a word in the regular language $(\mathsf{States} \cdot \Sigma)^* \cdot \mathsf{States}$.

Example 4

$(\mathtt{on}\,\mathtt{ok})\;\mathtt{switch}\;(\overline{\mathtt{on}}\,\mathtt{ok})\;\mathtt{switch}\;(\mathtt{on}\,\mathtt{ok})\;\mathtt{break}\;(\mathtt{on}\,\overline{\mathtt{ok}})$

Trace semantics – 2

Definition 5

A trace model for programs is $\mathsf{tra} : \Sigma \to Tr$ *where*

$$\mathsf{tra}(\mathtt{a}) \subseteq \{S\mathtt{a}T \mid S, T \textit{ states}\}$$

The canonical trace model is $\mathsf{can} : \mathtt{a} \mapsto \{S\mathtt{a}T \mid S, T \textit{ states}\}$.

Fusion product: partial function $\diamond : Tr \times Tr \to Tr$

$$xS \diamond Ty = \begin{cases} xSy & S = T \\ \text{undefined} & S \neq T \end{cases}$$

Lifted to sets of traces: $K \diamond L = \{w \diamond u \mid w \in K \ \ \& \ \ u \in L\}$. We define $K^0 := \mathsf{States}$, $K^{n+1} = K^n \diamond K$, and $K^* = \bigcup_{n \geq 0} K^n$.

Trace semantics – 3

Definition 6

Given tra, *we define* $[\![-]\!]_{\mathsf{tra}} : Fm \cup Pr \to 2^{Tr}$ *as follows:*

- $[\![B]\!]_{\mathsf{tra}} = \{S \mid S \vDash B\}$ *and* $[\![\mathsf{a}]\!]_{\mathsf{tra}} = \mathsf{tra}(\mathsf{a})$
- $[\![\mathbf{skip}]\!]_{\mathsf{tra}} = \mathsf{States} \quad [\![\mathbf{abort}]\!]_{\mathsf{tra}} = \emptyset$
- $[\![E; F]\!]_{\mathsf{tra}} = [\![E]\!]_{\mathsf{tra}} \diamond [\![F]\!]_{\mathsf{tra}}$
- $[\![\mathbf{if}\ B\ \mathbf{then}\ E\ \mathbf{else}\ F]\!]_{\mathsf{tra}} = ([\![B]\!]_{\mathsf{tra}} \diamond [\![E]\!]_{\mathsf{tra}}) \cup ([\![\neg B]\!]_{\mathsf{tra}} \diamond [\![F]\!]_{\mathsf{tra}})$
- $[\![\mathbf{while}\ B\ \mathbf{do}\ E]\!]_{\mathsf{tra}} = ([\![B]\!]_{\mathsf{tra}} \diamond [\![E]\!]_{\mathsf{tra}})^* \diamond [\![\neg B]\!]_{\mathsf{tra}}$

We denote $[\![-]\!]_{\mathsf{can}}$ *simply as* $[\![-]\!]$.

Trace semantics – 4

Example 5

An example trace model for $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\Pi = \{\mathtt{p}, \mathtt{q}\}$:

- $[\![\mathtt{p}]\!] = \{\mathtt{pq}, \mathtt{p\bar{q}}\}$, $[\![\mathtt{q}]\!] = \{\mathtt{pq}, \mathtt{\bar{p}q}\}$
- $[\![\mathtt{a}]\!] = \{\mathtt{pqa\bar{p}q}, \mathtt{p\bar{q}a\bar{p}\bar{q}}, \mathtt{\bar{p}qapq}, \mathtt{\bar{p}\bar{q}ap\bar{q}}\}$, $[\![\mathtt{b}]\!] = \{\mathtt{pqbp\bar{q}}, \mathtt{p\bar{q}bpq}, \mathtt{\bar{p}qb\bar{p}\bar{q}}, \mathtt{\bar{p}\bar{q}b\bar{p}q}\}$
- $[\![\mathtt{a}; \mathtt{b}]\!] = \{\mathtt{pqa\bar{p}qb\bar{p}\bar{q}}, \mathtt{\bar{p}qapqbp\bar{q}}, \mathtt{p\bar{q}a\bar{p}\bar{q}b\bar{p}q}, \ldots\}$
- $[\![\mathbf{if}\ \mathtt{p}\ \mathbf{then}\ \mathtt{a}\ \mathbf{else}\ \mathtt{b}]\!] = \{\mathtt{pqa\bar{p}q}, \mathtt{p\bar{q}a\bar{p}\bar{q}}, \mathtt{\bar{p}qb\bar{p}\bar{q}}, \mathtt{\bar{p}\bar{q}b\bar{p}q}\}$
- $[\![\mathbf{while}\ \mathtt{p}\ \mathbf{do}\ \mathtt{a}]\!] = \{\mathtt{pqa\bar{p}q}, \mathtt{p\bar{q}a\bar{p}\bar{q}}, \mathtt{\bar{p}q}, \mathtt{\bar{p}\bar{q}}\}$, $[\![\mathbf{while}\ \mathtt{p}\ \mathbf{do}\ \mathtt{b}]\!] = \{\mathtt{\bar{p}q}, \mathtt{\bar{p}\bar{q}}\}$

Trace semantics – 5

Theorem 1

$E \equiv F$ *iff* $[\![E]\!] = [\![F]\!]$.

Proof (sketch). 1. For each $M = \langle X, \mathsf{sat}_M, \mathsf{rel}_M \rangle$, let $\hat{M} : Tr \to 2^{X \times X}$ such that

$$\hat{M}(S) = \bigcap_{\mathtt{p} \in S} [\![\mathtt{p}]\!]_M \qquad \hat{M}(w\mathtt{a}u) = \hat{M}(w) \circ \mathsf{rel}_M(\mathtt{a}) \circ \hat{M}(u)\,.$$

We denote $\hat{M}(K) = \bigcup_{w \in K} \hat{M}(w)$ and prove $[\![E]\!]_M = \hat{M}([\![E]\!])$ by induction on E.

2. Let $\mathsf{cay} : 2^{Tr} \to 2^{Tr \times Tr}$ where

$$\mathsf{cay}(L) = \{\langle w, w \diamond u \rangle \mid w \in Tr \ \ \& \ \ u \in L\}$$

The function cay is injective. We prove by induction on E that $[\![E]\!]_M = \mathsf{cay}\,([\![E]\!])$ for $M = \langle Tr, \mathsf{sat}_M, \mathsf{rel}_M \rangle$ where $\mathsf{sat}_M(\mathtt{p}) = \{w \mid \langle w, w \rangle \in \mathsf{cay}\,([\![\mathtt{p}]\!])\}$ and $\mathsf{rel}_M(\mathtt{a}) = \mathsf{cay}\,([\![\mathtt{a}]\!])$. □

Completeness – 1

Recall the GKAT axioms:

BA. $B = C$ valid in BA

U1. $E +_B E = E$

U2. $E +_B F = F +_{(\neg B)} E$

U3. $(E +_B F) +_C G = E +_{(B \wedge C)} (F +_C G)$

U4. $E +_B F = B; E +_B F$

U5. $(E +_B F); G = (E; G) +_B (F; G)$

S1. $E; (F; G) = (E; F); G$

S2. $0; E = 0$

S3. $E; 0 = 0$

S4. $1; E = E$

S5. $E; 1 = E$

W1. $E^{(B)} = E; E^{(B)} +_B 1$

W2. $(E +_C 1)^{(B)} = (C; E)^{(B)}$

W3. $\dfrac{G = E; G +_B F}{G = E^{(B)}; F}$ if $\mathcal{H}(E) \equiv \bot$

Definition 7

A program equation $E = F$ is provable iff it is derivable from the GKAT axioms (using equational logic). Notation: $\vdash E = F$.

Completeness – 2

Let GKAT+UA be the set of GKAT axioms extended with the Uniqueness Axiom of Smolka et al. 2020. We express by $\vdash_{\mathsf{UA}} E = F$ that $E = F$ is provable in GKAT+UA.

Theorem 2

$\vdash_{\mathsf{UA}} E = F$ *iff* $[\![E]\!] = [\![F]\!]$.

Proof is beyond the scope of these lectures; see Smolka et al. 2020. □

Open problem

Give a sound and complete axiomatization of relational equivalence using only standard equational and quasi-equational axioms.

Hoare completeness – 1

Partial correctness:

If B holds and E is executed, then C will hold upon termination of E

(notation: $\{B\}E\{C\}$).

Hoare logic:

$$\frac{}{\{B\}\mathbf{skip}\{B\}} \qquad \frac{}{\{B\}\mathbf{abort}\{\bot\}} \qquad \frac{\{B\}E\{C\}\ \ \{C\}F\{D\}}{\{B\}E;F\{D\}}$$

$$\frac{\{B \wedge C\}E\{D\}\ \ \{\neg B \wedge C\}F\{D\}}{\{C\}\mathbf{if}\ B\ \mathbf{then}\ E\ \mathbf{else}\ F\{D\}} \qquad \frac{\{B \wedge C\}E\{C\}}{\{C\}\mathbf{while}\ B\ \mathbf{do}\ E\{\neg B \wedge C\}}$$

$$\frac{B' \vDash B\ \ \{B\}E\{C\}\ \ C \vDash C'}{\{B'\}E\{C'\}}$$

Hoare completeness – 2

$\{B\}E\{C\}$ is satisfied in M (notation $M \models \{B\}E\{C\}$) iff $s \in \mathsf{sat}_M(B)$ and $\langle s,t\rangle \in \mathsf{rel}_M(E)$ imply that $t \in \mathsf{sat}_M(C)$. The following are equivalent:

1. $M \models \{B\}E\{C\}$
2. $[\![B;E;C]\!]_M = [\![B;E]\!]_M$
3. $[\![B;E;\bar{C}]\!]_M = [\![\bot]\!]_M$.

Theorem 3

$\vdash B;E;C = B;E$ *iff* $[\![B;E;C]\!] = [\![B;E]\!]$.

Proof (sketch). 1. Soundness: Induction on length of derivation. 2. Completeness: Structural induction on E. □

Exercises – 1

1.1 Formalise part (ii) of the motivating example.

1.2 Formalise the following two programs. Are they equivalent?

def $\texttt{GCD}_1(a, b)$:

while $a \neq b$ **do**
if $(a > b)$ **then**
$a := a - b$
else
$b := b - a$
end if
end while
`print` a

def $\texttt{GCD}_2(a, b)$:

if $a \neq b$ **then**
if $(a > b)$ **then**
$a := a - b$
else
$b := b - a$
end if
end if
while $a \neq b$ **do**
if $(a > b)$ **then**
$a := a - b$
else
$b := b - a$
end if
end while
`print` a

Exercises – 2

1.3 Give examples of pairs of programs you think are equivalent. Formalise them in the language of programs.

1.4 Prove Proposition 1.

1.5 Can you show that programs in Exercise 1.2 are equivalent by deriving the corresponding equation from the GKAT axioms?

1.6 In the trace model of Example 1.5, what is the interpretation of the programs $\mathbf{if}\ \mathtt{p}\ \mathbf{then}\ \mathtt{a}\ \mathbf{else}\ \mathtt{a};\mathtt{b}$ and $\mathbf{while}\ \mathtt{p}\ \mathbf{do}\ \mathtt{b};\mathtt{a}$?

1.7 Verify that the GKAT axioms are valid in the canonical trace model for $\Pi = \{\mathtt{p}, \mathtt{q}\}$ and $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

1.8 Show that the function cay defined in the proof of Theorem 1 is injective.

1.9 Finish the proof of Theorem 3.

Notes

- The “logic of programs” introduced in this handout is a version of Guarded Kleene Algebra With Tests; see (Smolka et al., 2020).
- It is a “propositional variant” of while programs; see (Hoare, 1969) and Ch. 3 of (Apt et al., 2009).
- Our proof of Theorem 1 derives from Kappé’s lecture notes (Kappé, 2022); the argument goes back to (Pratt, 1980).
- (A first-order version of) Hoare logic was introduced by Hoare (1969); the relation of its propositional version to a formalism related to ours is studied by Kozen (2000).
- Theorems 3 and 2 are established in (Smolka et al., 2020).
- The standard completeness problem is discussed in (Kappé et al., 2023).

References

- Krzysztof R. Apt, Frank S. de Boer, and Ernst-Rüdiger Olderog. *Verification of Sequential and Concurrent Programs*. Texts in Computer Science. Springer, 3rd edition, 2009.
- Charles Anthony R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12:576–580, 1969.
- Tobias Kappé. Kleene algebra. Course notes, University of Amsterdam, 2022.
- Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Logic*, 1(1):60–76, July 2000.
- Tobias Kappé, Todd Schmid, and Alexandra Silva. A Complete Inference System for Skip-free Guarded Kleene Algebra with Tests. In T. Wies, editor, *Programming Languages and Systems. ESOP 2023.*, pages 309–336. Springer Nature Switzerland, 2023.
- Vaughan Pratt. Dynamic algebras and the nature of induction. In *Proc. Twelfth Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 22–28. ACM, 1980.
- Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Verification of uninterpreted programs in nearly linear time. In *Proc. 47th ACM SIGPLAN Symp. Principles of Programming Languages (POPL'20)*, pages 61:1–28, New Orleans, January 2020. ACM.