Lecture Notes on
Dynamic Logic

15-316: Software Foundations of Security & Privacy
Matt Fredrikson

Lecture 4
January 21, 2024

1 Introduction

In the last lecture we introduced *propositional sequent calculus* because it is the foundation of most of the calculi we will investigate and because it helps us understand the basic principles underlying the sequent calculus. Let's summarize them:

- We define *sequents* $\Gamma \vdash \Delta$ with antecedents (assumptions) and succedents (goals).
- We give a *meaning* to sequents (also called a *semantics*): a sequent is *valid* if when all antecedents are true then some succedent is true.
- We also give *inference rules* to prove sequents formally in a bottom-up manner. These are divided into *right rules* (how to prove a succedent) and *left rules* (how to use an antecedent), and the identity rule connecting left and right.
- We connect the inference rules to the semantics by proving (mathematically, at the metalevel) that the sequent calculus is *sound* and *complete*:
  - A rule is sound if the validity of all premises implies the validity of the conclusion. If all rules are sound, the whole system of inference rules is sound.
  - A system of rules is *complete* if every valid sequent can be proved with them.

In the specific case of propositional sequent calculus we were able to prove completeness using the following steps:

- Every rule is *invertible* in the sense that if the conclusion is valid then all the premise must also be valid.

- Every rule is *reductive* in the sense that all premises are smaller than the conclusion by counting the number of connectives and logical constants.

As we move forward, we will have to give up some of these properties while retaining others.

In this lecture we begin the study of *dynamic logic*, which enriches the sequent calculus with formulas that talk directly about program execution. The key new ingredients are the modalities $[\alpha]Q$ and $\langle\alpha\rangle Q$, which express (partial) correctness properties of a program α with respect to a postcondition Q. We will define the syntax of our tiny language, give semantics to these modal formulas, and then derive proof rules for common program constructs such as assignments and conditionals.

2 Dynamic Logic: A Logic with Programs

In this course we use a tiny imperative programming language so we can be rigorous about the concepts we introduce. With it we illustrate and analyze the concepts that transfer to realistic languages. There will be small differences regarding the precise extent of the language as we move through various concepts. Here is our first cut. *Expressions* denote integers and are either constants, variables, or operators like addition and multiplication. Programs include variable assignment, sequential composition, conditionals, and loops. Formulas no longer have propositional variables (for simplicity), but we add comparisons between integers to the usual set of logical constants and connectives. New here are two kinds of formulas, $[\alpha]Q$ and $\langle\alpha\rangle Q$ that mention programs. We explain them below the table.

Variables	x, y, z		
Constants	c	$::=$	$\ldots, -1, 0, 1, \ldots$
Expressions	e	$::=$	$c \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid \ldots$
Programs	α, β	$::=$	$x := e \mid \alpha \,;\, \beta \mid \textbf{if } P \textbf{ then } \alpha \textbf{ else } \beta \mid \textbf{while } P\ \alpha$
Formulas	P, Q	$::=$	$e_1 \leq e_2 \mid e_1 = e_2 \mid \ldots$
		$\mid$	$P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid P \leftrightarrow Q \mid \neg P \mid \top \mid \bot$
		$\mid$	$\forall x. P(x) \mid \exists x. P(x)$
		$\mid$	$[\alpha]Q \mid \langle\alpha\rangle Q$

A *state* is a total map from variables to integer values. We use ω, μ, ν for states. A program represents a partial function from an initial state (called *prestate*) to a final state (called *poststate*). It is a partial function because loops may not terminate, so no final state may every be reached. The sequence of states that a program goes through is its *trace* as discussed above.

Characteristic of *dynamic logic* [Harel, 1979, Harel et al., 2000] are two modalities that mention programs:

- $[\alpha]Q$ (pronounced "*box alpha Q*") which is true if, starting in a prestate ω, the formula Q will be true in every poststate ν we can reach by executing program α.

- $\langle\alpha\rangle Q$ (pronounced "*diamond alpha Q*") which is true in a state ω if there is a poststate ν that we can reach by executing α in which Q is true.

We refer to Q in these formulas as a *postcondition*. These definitions are formulated to account for *nondeterministic* programs that may have multiple poststates for a given prestate. In our case of a deterministic language, this will be zero or one. Therefore, we can already see that $\langle\alpha\rangle Q$ should imply $[\alpha]Q$.

In the next lecture we introduce a rigorous *semantics* for dynamic logic (which, by necessity, also includes programs and expressions). For today, we instead try to derive some rules keeping in mind the informal semantics and our understanding how an imperative programming language executes.

3 Semantics of Formulas

The semantics of formulas in a given state must appeal to the meaning of expressions and the meaning of programs. Therefore the meanings of programs and formulas mutually depend on each other. We start with some simple cases.

$$\begin{array}{lll}
\omega \models e_1 \leq e_2 & \text{iff} & \omega[\![e_1]\!] \leq \omega[\![e_2]\!] \\
\omega \models e_1 = e_2 & \text{iff} & \omega[\![e_1]\!] = \omega[\![e_2]\!] \\
\\
\omega \models P \wedge Q & \text{iff} & \omega \models P \quad \text{and} \quad \omega \models Q \\
\omega \models P \vee Q & \text{iff} & \omega \models P \quad \text{or} \quad \omega \models Q \\
\omega \models P \rightarrow Q & \text{iff} & \omega \models P \quad \text{implies} \quad \omega \models Q \\
\omega \models \neg P & \text{iff} & \omega \not\models P \\
\omega \models P \leftrightarrow Q & \text{iff} & \omega \models P \quad \text{iff} \quad \omega \models Q \\
\omega \models \forall x.\, P(x) & \text{iff} & \omega[x \mapsto c] \models P(x) \quad \text{for every } c \in \mathbb{Z} \\
\omega \models \exists x.\, P(x) & \text{iff} & \omega[x \mapsto c] \models P(x) \quad \text{for some } c \in \mathbb{Z}
\end{array}$$

Here $\omega[x \mapsto c]$ denotes the state obtained from ω by mapping x to c and leaving all other variables unchanged.

For programs, we have to recall the informal definition from earlier. $[\alpha]Q$ is true if Q is true in *every* poststate of α. Because a nonterminating program does not have a poststate, this is a statement about the *partial correctness* of the program α. Conversely, $\langle\alpha\rangle Q$ is true if Q is true in *some* poststate of α.

$$\begin{array}{lll}
\omega \models [\alpha]Q & \text{iff} & \text{for every } \nu \text{ with } \omega[\![\alpha]\!]\nu \text{ we have } \nu \models Q \\
\omega \models \langle\alpha\rangle Q & \text{iff} & \text{there is a } \nu \text{ with } \omega[\![\alpha]\!]\nu \text{ and } \nu \models Q
\end{array}$$

Now we say P is *valid* (written as $\models P$) if $\omega \models P$ for every state ω. (Recall that all states are defined on all variables, so this is well-defined.)

A sequent $P_1, \ldots, P_n \vdash Q_1, \ldots, Q_m$ is valid if for every state ω, whenever for all antecedents P_i we have $\omega \models P_i$ then for some succedent Q_j we have $\omega \models Q_j$.

4 Rules and Axioms

Our goal is ultimately to find proof rules for Dynamic Logic formulas containing boxes and diamonds. We'll start with a general observation: if P and Q are equivalent (i.e. $P \leftrightarrow Q$ is valid), then rules such as

$$\frac{\Gamma \vdash Q, \Delta}{\Gamma \vdash P, \Delta} \qquad \frac{\Gamma, Q \vdash \Delta}{\Gamma, P \vdash \Delta}$$

are **both sound and invertible**. That's because P and Q are always either both false or both true, regardless of the state because the bi-implication $P \leftrightarrow Q$ is valid. We call these valid formulas *axioms*, and we'll use this observation to identify axioms that we can use to derive sound and invertible proof rules.

In order to obtain good left and right rules from axioms we just have to make sure the rules are reductive. For example, the following equivalence is reductive in that it has the effect of making the program $\alpha\,;\,\beta$ smaller, breaking it into α and β that reside in separate boxes.

$$[\alpha\,;\,\beta]Q \leftrightarrow [\alpha]([\beta]Q)$$

Let's prove this. We start with the right-to-left direction, which implies the *soundness* of $[;]R$ and *invertibility* of $[;]L$. We set up the proof:

$\omega \models [\alpha]([\beta]Q)$ (assumption)
...
$\omega \models [\alpha\,;\,\beta]Q$ (to show)

We now walk through the proof in individual steps, narrowing the gap, sometimes from below and sometimes from above. Typically, one only presents the end result and the reader has to figure out how one might have obtained it.

By definition, the conclusion holds if for every state ν such that $\omega[\![\alpha\,;\,\beta]\!]\nu$ we have $\nu \models Q$. Now our proof state is (highlighting the new parts in blue):

$\omega \models [\alpha]([\beta]Q)$ (1, assumption)
$\omega[\![\alpha\,;\,\beta]\!]\nu$ for some ν (2, assumption)
...
$\nu \models Q$ (to show)
$\omega \models [\alpha\,;\,\beta]Q$ (by defn. of $\models$)

By definition, assumption 2 is true if there is some intermediate state μ such that $\omega[\![\alpha]\!]\mu$ and $\mu[\![\beta]\!]\nu$. Let's write this into the proof as well.

$\omega \models [\alpha]([\beta]Q)$ (1, assumption)
$\omega[\![\alpha\,;\,\beta]\!]\nu$ for some ν (2, assumption)
$\omega[\![\alpha]\!]\mu$ and $\mu[\![\beta]\!]\nu$ for some μ (3, from 2 by defn. of $[\![-]\!]$)
...
$\nu \models Q$ (to show)
$\omega \models [\alpha\,;\,\beta]Q$ (by defn. of $\models$)

Next: from assumption 1 and the fact that $\omega[\![\alpha]\!]\mu$ we can conclude that $\mu \models [\beta]Q$, again just by the definition of $\models$.

$\omega \models [\alpha]([\beta]Q)$	(1, assumption)
$\omega[\![\alpha\,;\,\beta]\!]\nu$ for some ν	(2, assumption)
$\omega[\![\alpha]\!]\mu$ and $\mu[\![\beta]\!]\nu$ for some μ	(3, from 2 by defn. of $[\![-]\!]$)
$\mu \models [\beta]Q$	(4, from 1 and 3(a) by defn. of $\models$)
...	
$\nu \models Q$	(to show)
$\omega \models [\alpha\,;\,\beta]Q$	(by defn. of $\models$)

Now we use the same argument knowing that $\mu[\![\beta]\!]\nu$ and $\mu \models [\beta]Q$ to conclude that $\nu \models Q$. But that's what we needed to show!

$\omega \models [\alpha]([\beta]Q)$	(1, assumption)
$\omega[\![\alpha\,;\,\beta]\!]\nu$ for some ν	(2, assumption)
$\omega[\![\alpha]\!]\mu$ and $\mu[\![\beta]\!]\nu$ for some μ	(3, by defn. of $[\![-]\!]$ from 2)
$\mu \models [\beta]Q$	(4, from 1 and 3(a) by defn. of $\models$)
$\nu \models Q$	(5, from 4 and 3(b) by defn. of $\models$)
$\omega \models [\alpha\,;\,\beta]Q$	(from 5 and 2 by defn. of $\models$)

We see the proof is actually quite straightforward. We just have to carefully unwind the definitions.

Here is the proof in the other direction.[1] We set up:

$\omega \models [\alpha\,;\,\beta]Q$	(1, assumption)
...	
$\omega \models [\alpha]([\beta]Q)$	(to show)

We show the filled-in proof. You can probably walk through it in the order we made the deductions.

$\omega \models [\alpha\,;\,\beta]Q$	(1, assumption)
$\omega[\![\alpha]\!]\mu$ for some μ	(2, assumption)
$\mu[\![\beta]\!]\nu$ for some ν	(3, assumption)
$\omega[\![\alpha\,;\,\beta]\!]\nu$	(4, from 2 and 3 by defn. of $[\![-]\!]$
$\nu \models Q$	(5, from 1 and 4 by defn of $\models$)
$\mu \models [\beta]Q$	(6, from 5 and 3 by defn of $\models$)
$\omega \models [\alpha]([\beta]Q)$	(7, from 6 and 2 by defn of $\models$)

Now that we know that $[\alpha\,;\,\beta]Q \leftrightarrow [\alpha]([\beta]Q)$ is valid, we can derive left and right rules for reasoning about sequential composition.

$$\frac{\Gamma \vdash [\alpha]([\beta]Q), \Delta}{\Gamma \vdash [\alpha\,;\,\beta]Q, \Delta}\,[;]R \qquad \frac{\Gamma, [\alpha]([\beta]Q) \vdash \Delta}{\Gamma, [\alpha\,;\,\beta]Q \vdash \Delta}\,[;]L$$

[1]not shown in lecture

5 Some Axioms for Dynamic Logic

Based on the insights from the previous section and the rules for programs in dynamic logic, we can conjecture the following axioms. And, indeed, they are all valid, even though we don't show the proofs. We write the postfix "A" to indicate that what we are naming is not a rule but an axiom.

$[:=]A$	$[x := e]Q(x) \leftrightarrow \forall x'.x' = e \rightarrow Q(x')$ (x' not in e or $Q(x)$)
$[;]A$	$[\alpha \,;\, \beta]Q \leftrightarrow [\alpha]([\beta]Q)$
$[\mathbf{if}]A$	$[\mathbf{if}\ P\ \mathbf{then}\ \alpha\ \mathbf{else}\ \beta]Q \leftrightarrow (P \rightarrow [\alpha]Q) \wedge (\neg P \rightarrow [\beta]Q)$
$[\mathbf{unfold}]A$	$[\mathbf{while}\ P\ \alpha] \leftrightarrow (P \rightarrow [\alpha][\mathbf{while}\ P\ \alpha]Q) \wedge (\neg P \rightarrow Q)$

Because these axioms are valid biconditionals, we obtain correct left and right rules for the sequent calculus. Note that directly translating these axioms into rules may be somewhat unsatisfactory. For example, the rules for [**unfold**] would not be reductive, and the rules for [:=] and [**if**] introduce complicated formulas. In the next lecture, we'll see how this can be addressed.

For now, let's take a closer look at the $[:=]A$ axiom, which may not seem obvious at first glance. The first instinct might be the following axiom for assignment

$$[x \leftarrow e]P \leftrightarrow (x = e \rightarrow P) \qquad \text{(WRONG)}$$

However, this is *not valid* and could therefore lead to unsound reasoning. For example, the following is certainly not valid

$$\not\models x = 3 \rightarrow [x \leftarrow x + 1](x = 17)$$

since computing $x \leftarrow x + 1$ will set x to 4 but the postcondition requires x to be 17. With the wrong axiom we could prove

$$(x = 3 \rightarrow [x \leftarrow x + 1]x = 17) \leftrightarrow (x = 3 \rightarrow ((x = x + 1) \rightarrow x = 17))$$

and the right-hand side is true since $x = x + 1$ is contradictory.

There are two ways out: one is to *carefully* substitute e for x with a so-called *uniform substitution*. The other is to *rename* the variable x, something we also did when generating a verification condition for a loop. This is handled by quantification over a fresh variable that does not occur in P. We write $P(x)$ for a formula P with (possible) occurrences of x, and then $P(x')$ for the result of renaming all occurrences of x to x'. Then our axiom becomes

$$[x \leftarrow e]P(x) \leftrightarrow (\forall x'.\, x' = e \rightarrow P(x'))$$

It is important for soundness that x' is a variable that does not already occur in e or $P(x)$. We often refer to this as a "fresh variable".

Our example no longer gives us a contradiction, because

$$(x = 3 \rightarrow [x \leftarrow x + 1]x = 17) \leftrightarrow (x = 3 \rightarrow \forall x'.\, x' = x + 1 \rightarrow x' = 17)$$

is false as it should be.

6 Addendum: Soundness of the Assignment Axiom

In this addendum we prove that the assignment axiom $[{:=}]A$ is valid. Recall the statement (with x' *fresh*, i.e. x' does not occur in e or $Q(x)$):

$$[x := e]Q(x) \leftrightarrow \forall x'.\, x' = e \rightarrow Q(x')$$

We start with the left-to-right direction. We set up the proof:

$\omega \models [x := e]Q(x)$ (1, assumption)
...
$\omega \models \forall x'.\, x' = e \rightarrow Q(x')$ (to show)

Now we proceed like before.

$\omega \models [x := e]Q(x)$ (1, assumption)
$\omega[\![x := e]\!]\nu$ for $\nu = \omega[x \mapsto \omega[\![e]\!]]$ (2, by defn. of $[\![{:=}]\!]$)
$\omega[x \mapsto \omega[\![e]\!]] \models Q(x)$ (3, by 2 and defn of $[\cdot]$)
$\omega[x' \mapsto \omega[\![e]\!]] \models Q(x')$ for fresh x' (4, by 3 and x' fresh)
$\omega[x' \mapsto c] \models x' = e \rightarrow Q(x')$ for every $c \in \mathbb{Z}$ (5, by 4 and defn. $\rightarrow$)
$\omega \models \forall x'.\, x' = e \rightarrow Q(x')$ (6, by 5 and defn. of $\forall$)

In the above, $Q(x')$ is the result of renaming the free occurrences of x in Q to the fresh variable x'. Since $c = \omega[\![e]\!]$, the value of x in ν equals the value of x' in $\omega[x' \mapsto c]$, and all other variables agree with ω; therefore $\nu \models Q(x)$ implies $\omega[x' \mapsto c] \models Q(x')$. This is exactly what we needed, completing the left-to-right direction.

Here is the proof in the other direction.

$\omega \models \forall x'.\, x' = e \rightarrow Q(x')$ (1, assumption)
$\omega[x' \mapsto c] \models x' = e \rightarrow Q(x')$ for all $c \in \mathbb{Z}$ (2, by defn. $\forall$)
$\omega[x' \mapsto \omega[\![e]\!]] \models Q(x')$ (3, by 2 and defn. of $\rightarrow$)
$\omega[x \mapsto \omega[\![e]\!]] \models Q(x)$ (4, by 3 and x' fresh)
$\omega[\![x := e]\!]\nu$ for any $\nu = \omega[x \mapsto \omega[\![e]\!]]$ (5, by defn. of $[\![{:=}]\!]$)
$\omega \models [x := e]Q(x)$ (6, from 4 and 5 by defn. of $[\cdot]$)

References

David Harel. *First-Order Dynamic Logic.* Springer LNCS 68, 1979. 136 pp.

David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic.* MIT Press, 2000. 476 pp.