**15-414: Bug Catching: Automated Program Verification**

# Lecture Notes on Diamonds and Total Correctness

Matt Fredrikson Ruben Martins

Carnegie Mellon University
Lecture 8

## 1 Introduction

The previous lectures gave us a good understanding of how to reason about $[\cdot]$ properties of programs. We saw how to decompose in logic properties of the form $[\alpha;\beta]P$ and $[\mathtt{if}(Q)\,\alpha\,\mathtt{else}\,\beta]P$ etc. That is all good and useful, but we need to remember that $[\alpha]P$ means that $P$ holds after all runs of program $\alpha$. Since while programs $\alpha$ are deterministic and, thus, have at most one run, the formula $[\alpha]P$, in particular, means that:

*if* $\alpha$ terminates, then $P$ holds in the final state

But formula $[\alpha]P$ says nothing at all about whether the program terminates. Because it makes no statement about termination, this notion is also called *partial correctness* of program $\alpha$ with respect to postcondition $P$.

This is what the diamond modality $\langle\cdot\rangle$ is good for. The formula $\langle\alpha\rangle P$ says that there is a run of program $\alpha$ that reaches a final state in which postcondition $P$ is true. Since while programs $\alpha$ have at most one run, the formula $\langle\alpha\rangle P$, in particular, means that:

$\alpha$ terminates *and* $P$ holds in the final state.

Because it guarantees both a correct result and termination, this notion is also called *total correctness* of program $\alpha$ with respect to postcondition $P$.

Total correctness is the stronger notion compared to partial correctness, because it says that a program not only gives the right answer upon termination but also actually stops in finite time. Consequently, in order to understand total correctness and how to reason about total correctness of while programs, we investigate diamond modalities [HKT00, Pla17b].

## 2 Diamond Axioms for Programs

Our approach to understanding programs with logic still is to design one reasoning principle for each program operator that describes its effect in logic with simpler logical operators. If we succeed doing that for every operator that a program can have, then we will understand even the most complicated programs just by repeatedly making use of the respective logical reasoning principles. Only this time it will be the combination of diamond modalities and the respective program operators we worry about.

### 2.1 Assignments

The easiest case to look into is what we need to prove in order to show the formula $\langle x := e\rangle p(x)$, which expresses that the assignment $x := e$ terminates and the formula $p(x)$ holds after the assignment $x := e$ that assigns the value of term $e$ to variable $x$. How could we reduce this to another logical formula that is simpler?

Obviously assignments always terminate, because their only effect is to change the value of one variable which succeeds in (very) finite time. If we want to show that the formula $p(x)$ holds after assigning the new value $e$ to variable $x$ then we might as well show $p(e)$ right away. And, in fact, $p$ is true of $x$ after assigning $e$ to $x$ if and only if $p$ is true of its new value $e$. That is, the formula $\langle x := e\rangle p(x)$ is equivalent to the formula $p(e)$. We capture this argument once and for all in the assignment axiom $\langle:=\rangle$:

$$(\langle:=\rangle)\quad \langle x := e\rangle p(x) \leftrightarrow p(e)$$

In the assignment axiom $\langle:=\rangle$, the formula $p(e)$ has the term $e$ everywhere in place of where the formula $p(x)$ has the variable $x$. Of course, it is important for this substitution of $e$ for $x$ to avoid capture of variables and not make any replacements under the scope of a quantifier or modality binding an affected variable [Pla17a]. For example, the following formula is an instance of $\langle:=\rangle$:

$$\langle x := x^2-1\rangle x(x+1) \geq x+y \leftrightarrow (x^2-1)(x^2-1+1) \geq (x^2-1)+y$$

But the following is not because it would capture the replacement $y$ that is used for $x$:

$$\langle x := y\rangle (x \geq 0 \land \forall y\, (x \geq y)) \leftrightarrow (y \geq 0 \land \forall y\, (y \geq y))$$

Instead, if we first rename $\forall y$ to $\forall z$ then the substitution works:

$$\langle x := y\rangle (x \geq 0 \land \forall z\, (x \geq z)) \leftrightarrow (y \geq 0 \land \forall z\, (y \geq z))$$

Indeed by combining the $[:=]$ and $\langle:=\rangle$ axioms, one can also show that both modalities are equivalent for assignments:

$$[x := e]p(x) \leftrightarrow \langle x := e\rangle p(x)$$

This makes sense because an assignment always terminates and has exactly one successor, so modalities quantifying over all or one successor actually express the same fact for assignments. But such an equivalence will not generally hold for other program operators!

Recall one implication for deterministic programs, which shows that total correctness implies partial correctness.

**Lemma 1** (Deterministic program modality relation)**.** *Deterministic programs $\alpha$ make the following formula valid for all formulas $P$:*

$$\langle\alpha\rangle P \rightarrow [\alpha]P$$

There is no reason to believe both sides would be equivalent in general, because there are many programs that are partially correct just never terminate. In fact, can you find a program that is partially correct for all preconditions $A$ and postconditions $B$?

Finally, since box and diamond modalities are equivalent in the case of assignments, the equational form of an assignment proof rule not only works for boxes but also for diamonds:

$$(\langle{:=}\rangle_{=})\ \frac{\Gamma, y = e \vdash p(y), \Delta}{\Gamma \vdash \langle x := e\rangle p(x), \Delta} \quad (y \text{ new})$$

## 2.2 Conditionals

The next case we choose to look at is what we need to prove in order to show the formula $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$, which expresses that formula $P$ holds in some final state reached after running the if-then-else conditional $\texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta$ that runs program $\alpha$ if formula $Q$ is true and runs $\beta$ otherwise. And that this conditional program will indeed terminate. Of course, the if will terminate but the question is whether the resulting subprograms $\alpha$ and/or $\beta$ terminate. In order to understand it from a logical perspective, how could we express $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ in easier ways?

If $Q$ holds then $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ says that $P$ holds in some final state after running $\alpha$. If $Q$ does not hold then the same formula $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ says that $P$ holds in some final state after running $\beta$. It is easy to say with a logical formula that $P$ holds in some state after running $\alpha$, which is precisely what $\langle\alpha\rangle P$ is good for. Likewise $\langle\beta\rangle P$ directly expresses in logic that $P$ in some state after running $\beta$. Both of those formulas $\langle\alpha\rangle P$ as well as $\langle\beta\rangle P$ are simpler than the original formula $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$. But, of course, they express something else, because the program $\texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta$ only runs the respective programs conditionally depending on the truth-value of $Q$.

Yet, there is a way of expressing $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ in logic in easier ways with the help of other logical operators. Implications are perfect at expressing the conditions that an if-then statement states in a program. Indeed, if $Q$ holds then $\langle\alpha\rangle P$ needs to be true and if $Q$ does not hold then $\langle\beta\rangle P$ for $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ to hold. Indeed, $\langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P$ is true if and only if $(Q \rightarrow \langle\alpha\rangle P) \wedge (\neg Q \rightarrow \langle\beta\rangle P)$ is true. This is the diamond formulation of the if-then-else axiom $\langle\text{if}\rangle$:

$$(\langle\text{if}\rangle)\ \langle \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle P) \wedge (\neg Q \rightarrow \langle\beta\rangle P)$$

This axiom tells us everything we need to know about correct termination of if-then-else statements. When using the equivalence $\langle\text{if}\rangle$ from left to right, we can use it to simplify every question about an if-then-else statement of the form $\langle \texttt{if}(Q)\, \alpha\, \texttt{else}\, \beta\rangle P$ by a corresponding structurally simpler formula $(Q \to \langle\alpha\rangle P) \land (\neg Q \langle\beta\rangle P)$ that does not use the if-then-else statement any more but is logically equivalent. The axiom $\langle\text{if}\rangle$ achieves the same kind of compositionality that axiom [if] achieves, just for diamond modalities of if-then-else statements. The $\langle\text{if}\rangle$ axiom will enable us, for example to conclude this equivalence:

$$\langle \texttt{if}(x\geq0)\, y:=x\, \texttt{else}\, y:=-x\rangle y=|x| \leftrightarrow (x\geq0 \to \langle y:=x\rangle y=|x|) \land (\neg x\geq0 \to \langle y:=-x\rangle y=|x|)$$

This formula uses $|x|$ as notation for the absolute value of $x$.

Since axiom $\langle\text{if}\rangle$ justifies this equivalence, we will be able to reduce a question about whether its left hand side is valid with axiom $\langle\text{if}\rangle$ to the question whether its corresponding right hand side is valid.

$$
{\scriptstyle\langle\text{if}\rangle}\dfrac{{\scriptstyle\land\text{R}}\dfrac{{\scriptstyle\to\text{R}}\dfrac{{\scriptstyle\langle:=\rangle}\dfrac{{\scriptstyle\mathbb{Z}}\dfrac{*}{x\geq0 \vdash x=|x|}}{x\geq0 \vdash \langle y:=x\rangle\, y=|x|}}{\vdash x\geq0 \to \langle y:=x\rangle\, y=|x|} \qquad {\scriptstyle\to\text{R}}\dfrac{{\scriptstyle\langle:=\rangle}\dfrac{{\scriptstyle\mathbb{Z}}\dfrac{*}{\neg x\geq0 \vdash -x=|x|}}{\neg x\geq0 \vdash \langle y:=-x\rangle\, y=|x|}}{\vdash \neg x\geq0 \to \langle y:=-x\rangle\, y=|x|}}{\vdash (x\geq0 \to \langle y:=x\rangle\, y=|x|) \land (\neg x\geq0 \to \langle y:=-x\rangle\, y=|x|)}}{\vdash \langle \texttt{if}(x\geq0)\, y:=x\, \texttt{else}\, y:=-x\rangle\, y=|x|}
$$

This proof shows validity of the following formula, which says that the given program is totally correct in implementing the absolute value function $|\cdot|$ from mathematics:

$$\langle \texttt{if}(x\geq0)\, y:=x\, \texttt{else}\, y:=-x\rangle\, y=|x|$$

As usual the proof is developed starting with the desired conclusion at the bottom and working with proof rules to the top as usual in sequent calculus.

## 2.3 Test

The test statement $?Q$ also checks a condition on the current state but has a different effect. It has no effect on the state if $Q$ is indeed true, but aborts and discards the execution if $Q$ is not true. In particular, in the latter case, the program did not terminate (rather it was aborted unsuccessfully). How can we express $\langle ?Q\rangle P$ in logic in structurally simpler ways?

The formula $\langle ?Q\rangle P$ is true iff formula $P$ holds in some final state after running the test $?Q$, which only even has a final state if $Q$ is true. Consequently $P$ holds in some final state after running the program $?Q$ iff postcondition $P$ is true and if the test $Q$ is. This is captured in the test axiom $\langle ?\rangle$:

$$(\langle ?\rangle)\quad \langle ?Q\rangle P \leftrightarrow (Q \land P)$$

Even if diamonds of deterministic programs imply their own boxes, the test axiom $\langle ? \rangle$ already alerts us to the fact that some programs are only partially correct but not totally correct. The following box formula is valid

$$[?x = 2]x \cdot x = 4$$

because by axiom $[?]$ it is equivalent to the valid formula $x = 2 \rightarrow x \cdot x = 4$. But the corresponding diamond formula is not valid:

$$\langle ?x = 2 \rangle x \cdot x = 4$$

because axiom $\langle ? \rangle$ makes it equivalent to $x = 2 \land x \cdot x = 4$ so $x = 2$, which is not valid but merely satisfiable.

## 2.4 Sequential Compositions

The next most pressing case to worry about are sequential compositions. So how can we equivalently express $\langle \alpha; \beta \rangle P$ in simpler logic without sequential compositions? This formula expresses that $P$ holds after some runs of $\alpha; \beta$, which first runs $\alpha$ and then runs $\beta$. How can this be expressed in an easier way in logic, again using just the subprograms $\alpha$ as well as $\beta$ of $\alpha; \beta$ then?

In order to express $\langle \alpha; \beta \rangle P$ what we need to say is that after some run of $\alpha$ it is the case that $P$ holds after some run of $\beta$. It is comparably easy to say that $P$ holds after some runs of $\beta$ just with the formula $\langle \beta \rangle P$. But where does this formula need to hold? After some runs of $\alpha$! In particular, all we need to say is that $\langle \beta \rangle P$ holds after some run of $\alpha$, which is exactly what the formula $\langle \alpha \rangle \langle \beta \rangle P$ says. This is the sequential composition axiom $\langle ; \rangle$ for diamonds:

$$(\langle ; \rangle) \quad \langle \alpha; \beta \rangle P \leftrightarrow \langle \alpha \rangle \langle \beta \rangle P$$

Indeed, after all runs of $\alpha; \beta$ does $P$ hold if and only if after all runs of $\alpha$ it is the case that after all runs of $\beta$ does $P$ hold.

This enables us to consider another absolute value function implementation that is partially correct but *not* totally correct! The following formula is easily seen to be valid:

$$[y := -x; ?y \geq 0]\, y = |x|$$

A proof of validity is easy from the box axioms and arithmetic facts about the absolute value function $|\cdot|$:

$$\cfrac{\cfrac{\cfrac{\cfrac{*}{\vdash -x \geq 0 \rightarrow -x = |x|}\ \mathbb{Z}}{\vdash [?{-x} \geq 0]\, -x = |x|}\ [?]}{\vdash [y := -x][?y \geq 0]\, y = |x|}\ [:=]}{\vdash [y := -x; ?y \geq 0]\, y = |x|}\ [;]$$

A corresponding attempt to prove total correctness will, however, fail:

$$
\langle ;\rangle\,\dfrac{\langle :=\rangle\,\dfrac{\langle ?\rangle\,\dfrac{\text{cut}\,\dfrac{\vdash -x \geq 0 \qquad \wedge\text{R}\,\dfrac{\text{id}\,\dfrac{*}{-x \geq 0 \vdash -x \geq 0} \qquad \mathbb{Z}\,\dfrac{*}{-x \geq 0 \vdash -x = |x|}}{-x \geq 0 \vdash -x \geq 0 \wedge -x = |x|}}{\vdash -x \geq 0 \wedge -x = |x|}}{\vdash \langle ?{-x} \geq 0\rangle\, -x = |x|}}{\vdash \langle y := -x\rangle \langle ?y \geq 0\rangle\, y = |x|}}{\vdash \langle y := -x; ?y \geq 0\rangle\, y = |x|}
$$

In fact, the total correctness property is not valid:

$$\langle y := -x; ?y \geq 0\rangle\, y = |x|$$

Only if the formula $-x \geq 0$ in the remaining open proof branch is assumed in the initial state is the total correctness formula valid:

$$x \leq 0 \rightarrow \langle y := -x; ?y \geq 0\rangle\, y = |x|$$

## 2.5 Loop the Loop

Unwinding a loop is also possible for diamond modalities with while loops. If $Q$ holds then $\langle \texttt{while}(Q)\, \alpha\rangle P$ runs $\alpha$ and then runs the while loop afterwards yet again. If $Q$ does not hold then the loop has no effect and just stops right away. That is why $\texttt{while}(Q)\, \alpha$ is equivalent to $\texttt{if}(Q)\, \{\alpha; \texttt{while}(Q)\, \alpha\}$, because both have no effect if $Q$ is false but repeat $\alpha$ as long as $Q$ is true. We can capture these thoughts in the following axiom:

$$(\langle\text{unwind}\rangle)\quad \langle \texttt{while}(Q)\, \alpha\rangle P \leftrightarrow \langle \texttt{if}(Q)\, \{\alpha; \texttt{while}(Q)\, \alpha\}\rangle P$$

Of course, this axiom has the same shortcoming that the unwinding axiom already had for box modalities. If we unwind a loop, there still is a loop to be taken care of, which is quite problematic for loops that do not have a fixed small finite number of iterations.

As in the case of box modalities, a similar axiom can be derived by already applying the $\langle\text{if}\rangle$ and $\langle;\rangle$ axioms on the right-hand side:

$$(\langle\text{unfold}\rangle)\quad \langle \texttt{while}(Q)\, \alpha\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle\langle \texttt{while}(Q)\, \alpha\rangle P) \wedge (\neg Q \rightarrow P)$$

The $\langle\text{unfold}\rangle$ axiom can be used to prove total correctness of a loopy program:

$$s := 1; \texttt{while}(x < 0)\, \{x := -x; s := s - 2\}$$

This program totally correctly implements the nonnegative sign function:

$$\text{sig}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$
{\scriptstyle\langle ;\rangle}\dfrac{{\scriptstyle\langle :=\rangle_=}\dfrac{{\scriptstyle\langle \text{unfold}\rangle}\dfrac{{\scriptstyle\land\text{R}}\dfrac{\dfrac{\dots}{s{=}1 \vdash x{<}0 \to \langle x := -x; s := s-2\rangle\langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)} \quad {\scriptstyle\mathbb{Z}}\dfrac{*}{s{=}1 \vdash \neg x{<}0 \to s{=}\text{sig}(x)}}{s = 1 \vdash (x{<}0 \to \langle x := -x; s := s-2\rangle\langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)) \land (\neg x{<}0 \to s{=}\text{sig}(x))}}{s = 1 \vdash \langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)}}{\vdash \langle s := 1\rangle\langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)}}{\vdash \langle s := 1; \texttt{while}(x < 0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)}
$$

The left branch continues as follows using that $x < 0 \to \langle x := -x\rangle x > 0$ is valid:

$$
{\scriptstyle\to\text{R}}\dfrac{{\scriptstyle\langle \text{unfold}\rangle}\dfrac{{\scriptstyle\langle ;\rangle}\dfrac{{\scriptstyle\langle :=\rangle}\dfrac{\dfrac{{\scriptstyle\langle :=\rangle}\dfrac{{\scriptstyle\mathbb{Z}}\dfrac{*}{s{=}1, x{<}0 \vdash \neg{-x}{<}0 \to s-2{=}\text{sig}(-x)}}{s{=}1, x{<}0 \vdash \langle s := s-2\rangle(\neg{-x}{<}0 \to s{=}\text{sig}(-x))}}{s{=}1, x{<}0 \vdash \langle s := s-2\rangle\big((-x{<}0 \to \dots) \land (\neg{-x}{<}0 \to s{=}\text{sig}(-x))\big)}}{s{=}1, x{<}0 \vdash \langle x := -x\rangle\langle s := s-2\rangle\big((x{<}0 \to \langle\dots\rangle\langle \texttt{while}(x{<}0)\ \dots\rangle\, s{=}\text{sig}(x)) \land (\neg x{<}0 \to s{=}\text{sig}(x))\big)}}{s{=}1, x{<}0 \vdash \langle x := -x; s := s-2\rangle\big((x{<}0 \to \langle\dots\rangle\langle \texttt{while}(x{<}0)\ \dots\rangle\, s{=}\text{sig}(x)) \land (\neg x{<}0 \to s{=}\text{sig}(x))\big)}}{s{=}1, x{<}0 \vdash \langle x := -x; s := s-2\rangle\langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)}}{s{=}1 \vdash x{<}0 \to \langle x := -x; s := s-2\rangle\langle \texttt{while}(x{<}0)\,\{x := -x; s := s-2\}\rangle\, s{=}\text{sig}(x)}
$$

In general, of course, unwinding is only enough if the loop terminates in at most some known fixed finite number of iterations. Would the program also work correctly for a corresponding sign function when replacing its loop guard by $x \leq 0$?

The above proof also has a minor subtlety in the occurrence of $\dots$. Can you spot it and suggest a way of solving it?

## 3 Soundness

The above axioms can again all be shown to be sound. We only show the proof of one axiom in order to leave you sufficiently many other axioms to practice soundness proofs on.

**Lemma 2.** *The sequential composition axiom ⟨;⟩ is sound, i.e. all its instances are valid:*

$$(\langle ;\rangle)\quad \langle \alpha;\beta\rangle P \leftrightarrow \langle\alpha\rangle\langle\beta\rangle P$$

*Proof.* Recall the semantics of sequential composition:

$$[\![\alpha;\beta]\!] = [\![\alpha]\!] \circ [\![\beta]\!] = \{(\omega,\nu) \ :\ (\omega,\mu) \in [\![\alpha]\!], (\mu,\nu) \in [\![\beta]\!]\}$$

In order to show that the formula $\langle\alpha;\beta\rangle P \leftrightarrow \langle\alpha\rangle\langle\beta\rangle P$ is valid, i.e. $\vDash \langle\alpha;\beta\rangle P \leftrightarrow \langle\alpha\rangle\langle\beta\rangle P$, consider any state $\omega$ and show that $\omega \models \langle\alpha;\beta\rangle P \leftrightarrow \langle\alpha\rangle\langle\beta\rangle P$. We prove this biimplication by separately proving both implications.

"$\leftarrow$" Assume the right hand side $\omega \models \langle\alpha\rangle\langle\beta\rangle P$ and show $\omega \models \langle\alpha;\beta\rangle P$. To show the latter, we need to show that there is a state $\nu$ with $(\omega,\nu) \in [\![\alpha;\beta]\!]$ for which $\nu \models P$.

By the semantics of sequential composition, $(\omega, \nu) \in [\![\alpha; \beta]\!]$ iff there is a state $\mu$ such that $(\omega, \mu) \in [\![\alpha]\!]$ and $(\mu, \nu) \in [\![\beta]\!]$. The assumption implies that there is a state $\mu$ with $(\omega, \mu) \in [\![\alpha]\!]$ such that $\mu \models \langle\beta\rangle P$. The latter implies that there is a $\nu$ with $(\mu, \nu) \in [\![\beta]\!]$ such that $\nu \models P$. Thus, $\nu \models P$ and, as desired, $(\omega, \nu) \in [\![\alpha; \beta]\!]$, because $(\omega, \mu) \in [\![\alpha]\!]$ and $(\mu, \nu) \in [\![\beta]\!]$. Hence $\omega \models \langle\alpha; \beta\rangle P$.

"$\leftarrow$" Conversely, assume the left hand side $\omega \models \langle\alpha; \beta\rangle P$ and show $\omega \models \langle\alpha\rangle\langle\beta\rangle P$. Consequently, there is a state $\nu$ such that $(\omega, \nu) \in [\![\alpha; \beta]\!]$ and $\nu \models P$. Now $(\omega, \mu) \in [\![\alpha]\!]$ and $(\mu, \nu) \in [\![\beta]\!]$ iff $(\omega, \nu) \in [\![\alpha; \beta]\!]$ by the semantics of sequential composition. Hence, there is a state $\mu$ such that $(\omega, \mu) \in [\![\alpha]\!]$ and $\mu \models \langle\beta\rangle P$. Thus, $\omega \models \langle\alpha\rangle\langle\beta\rangle P$. □

**Lemma 3.** *The following axiom is a derived axiom, so can be proved from the other axioms in sequent calculus, and is, thus, sound:*

$$(\langle unfold\rangle)\ \langle \mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle\langle\mathtt{while}(Q)\,\alpha\rangle P) \wedge (\neg Q \rightarrow P)$$

*Proof.* The axiom ⟨unfold⟩ can be derived by using other axioms (from right to left):

$$\cfrac{\cfrac{\cfrac{*}{\vdash \langle\mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow \langle\mathtt{if}(Q)\,\{\alpha; \mathtt{while}(Q)\,\alpha\}\rangle P}\langle\text{unwind}\rangle}{\vdash \langle\mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow (Q \rightarrow \langle\alpha; \mathtt{while}(Q)\,\alpha\rangle P) \wedge (\neg Q \rightarrow P)}\langle\text{if}\rangle}{\vdash \langle\mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle\langle\mathtt{while}(Q)\,\alpha\rangle P) \wedge (\neg Q \rightarrow P)}\langle;\rangle$$

□

## 4 Summary

The axioms introduced in this lecture are summarize in Fig. 1. These axioms handle all total correctness properties of programs that only consist of assignment, conditionals, sequential compositions, and tests. But as soon as while loops are involved, things get more complicated, since the unwinding-based axioms help for finite bounded loops.

## References

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Pla17a] 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.

[Pla17b] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Switzerland, 2017. URL: http://www.springer.com/978-3-319-63587-3.

$$
\begin{aligned}
&(\langle{:=}\rangle) && \langle x := e\rangle p(x) \leftrightarrow p(e) \\
&(\langle ?\rangle) && \langle ?Q\rangle P \leftrightarrow (Q \land P) \\
&(\langle \mathrm{if}\rangle) && \langle \mathtt{if}(Q)\,\alpha\,\mathtt{else}\,\beta\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle P) \land (\neg Q \rightarrow \langle\beta\rangle P) \\
&(\langle ;\rangle) && \langle \alpha;\beta\rangle P \leftrightarrow \langle\alpha\rangle\langle\beta\rangle P \\
&(\langle \mathrm{unwind}\rangle) && \langle \mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow \langle \mathtt{if}(Q)\,\{\alpha;\mathtt{while}(Q)\,\alpha\}\rangle P \\
&(\langle \mathrm{unfold}\rangle) && \langle \mathtt{while}(Q)\,\alpha\rangle P \leftrightarrow (Q \rightarrow \langle\alpha\rangle\langle \mathtt{while}(Q)\,\alpha\rangle P) \land (\neg Q \rightarrow P)
\end{aligned}
$$

Figure 1: Diamond axioms of the day