Proving program termination

Byron Cook

1
Foundations

Our goal in this book is to build sofware tools that automatically search for proofs of program termination in mathematical logic. However, before delving directly into strategies for automation, we must first introduce some notation and establish a basic foundation in the areas of program semantics, logic and set theory. We must also discuss how programs can be proved terminating using manual techniques. The concepts and notation introduced in this chapter will be used throughout the remainder of the book.

1.1 Program termination and well-founded relations

For the purpose of this book it is convenient to think of the text of a computer program as representing a relation that specifies the possible transitions that the program can make between configurations during execution. We call this the program's *transition relation.* Program executions can be thought of as traversals starting from a starting configuration and then moving from configuration to configuration as allowed by the transition relation. A program is called *terminating* if all the executions allowed by the transition relation are finite. We call a program *non-terminating* if the transition relation allows for at least one infinite execution.

Treating programs as relations is conveinant for our purpose, as in this setting proving program termination is equivliant to proving the program's transition relation *well-founded*— thus giving us access to the numerous well established techniques from mathematical logic used to establish well-foundedness.

In the next few sections we define some notation, discuss our representation for program configurations, and give some basic results related

to to relations and well-foundedness. In a later section we will define a representation for computer programs and discuss manual methods for proving termination.

1.2 Program states, sets and relations

We will assume that a set of possible program variables, VAR, is infinite, and formed of a subset of the possible strings expressed in sans-serif font (*e.g.* $\mathsf{x} \in \text{VAR}$). Let $\boxed{\text{VAR}}$ be the set of variables drawn from VAR and then boxed (*e.g.* if $\mathsf{x} \in \text{VAR}$, then $\boxed{\mathsf{x}} \in \boxed{\text{VAR}}$). These boxed variables are usually used when two distinct sets of variables are needed. We will assume that, on a set V, $\boxed{V}$ always defines a one-to-one and onto mapping. We also assume that $\boxed{-}^{-1}$ is the inverse of $\boxed{-}$. That is, if $v = \boxed{\mathsf{x}}$, then $\boxed{v}^{-1} = \mathsf{x}$. We will use the symbol $\boxed{-}$ to represent the function that boxes variables: $\boxed{-}\,(x) \triangleq \boxed{x}$.

The set of possible program values, VAL, will be an under-specified set of values which can include $\mathbb{Z}$ and other arithmetic constants.

Throughout this book we will be concentrating on program configurations. We call these configurations *states*. States will be encoded as partial finite mappings from variables to values, $\text{VAR} \xrightarrow{fin} \text{VAL}$. For example $\{\mathsf{x} \mapsto 5, \mathsf{y} \mapsto -2\}$ is the state in which $\mathsf{x} = 5$, $\mathsf{y} = -2$, and undefined for other variables. Let $\mathcal{S}$ be the set of all such mappings $\text{VAR} \xrightarrow{fin} \text{VAL}$. The observant reader may have noticed that our formulation of program states does not currently allow for programs with data structures—this will come in later chapters.

Definition 1 (Relational application, composition, closure) Assume that $R \subseteq S \times T$ and $X \subseteq S$, we define the image of R on X (notationally, $R(X)$) as:

$$R(X) \triangleq \{y \mid x \in X \wedge (x, y) \in R\}$$

Note that $R(X) \subseteq T$. If $a \in S$ and $R \subseteq S \times T$ then we define $R(a) \triangleq R(\{a\})$. Let ; be relational composition where

$$R; Q \triangleq \{(a, b) \mid \exists c.\ (a, c) \in R \wedge (c, b) \in Q\}$$

When $k > 0$, $R^k \triangleq R; R^{k-1}$, and $R^0 \triangleq \{(a, b) \mid a = b\}$. The non-reflexive and reflexive transitive closure of R are defined respectively:

$$R^+ \triangleq \{(a, b) \mid \exists n > 0. (a, b) \in R^n\}$$

$$R^* \triangleq \{(a,b) \mid \exists n \geq 0.(a,b) \in R^n\}$$

We define relational inverse as

$$R^{-1} \triangleq \{(a,b) \mid (b,a) \in R\}$$

If Q is a set of states (*i.e.* $Q \subseteq \mathcal{S}$) then $\neg Q \triangleq \mathcal{S} - Q$

Definition 2 ($\overleftarrow{-}$, $\overrightarrow{-}$) Assume that $R \subseteq A \times B$. We define left projection $\overleftarrow{R} \subseteq A$ as

$$\overleftarrow{R} \triangleq \{a \mid \exists b.\ (a,b) \in R\}$$

We define right projection $\overrightarrow{R} \subseteq B$ as

$$\overrightarrow{R} \triangleq \{b \mid \exists a.\ (a,b) \in R\}$$

Definition 3 ($\underline{-}$) We define the *lifting of R via a function f*, notationally $\underline{R}_f$, to be:

$$\underline{R}_f \triangleq \{(s,t) \mid (f(s), f(t)) \in R\}$$

1.3 Symbolic formulae, sets and relations

When describing sets of states we will—depending on the context—use one of two possible representations: symbolic formulae and explicit sets. For example, the symbolic formula $\mathsf{x} > 0$ will represent the set of states $\{s \mid s(\mathsf{x}) > 0\}$. Formally, we define a semantic meaning $[\![-]\!]$ from formulae to the sets that that represent (*i.e.*, $[\![\mathsf{x} > 0]\!] = \{s \mid s(\mathsf{x}) > 0\}$).

Definition 4 ($[\![-]\!]$) Assume a formula f and a function g whose domain is the variables of f. We define $f[g]$ as the substitution of variables in f via g. We define the meaning of f as

$$[\![f]\!] = \{s \mid f[s]\}$$

Example 1 Consider the formula $\mathsf{x} > 0$. This formula represents the set of states

$$\begin{array}{rcl} [\![\mathsf{x} > 0]\!] & = & \{s \mid (\mathsf{x} > 0)[s]\} \\ & = & \{s \mid s(\mathsf{x}) > 0\} \end{array}$$

Now consider the state $\{\mathsf{x} \mapsto 5, \mathsf{y} \mapsto -2\}$. This state is in $[\![\mathsf{x} > 0]\!]$, because

$$(\mathsf{x} > 0)(\{\mathsf{x} \mapsto 5, \mathsf{y} \mapsto -2\}) \Leftrightarrow 5 > 0 \Leftrightarrow \mathsf{true}$$

The state $\{\mathsf{x} \mapsto 0, \mathsf{z} \mapsto 2\}$ is not in $[\![\mathsf{x} > 0]\!]$, as

$$(\mathsf{x} > 0)(\{\mathsf{x} \mapsto 0, \mathsf{z} \mapsto 2\}) \Leftrightarrow 0 > 0 \Leftrightarrow \mathsf{false}$$

$\star$

When discussing relations over states we will, again, use either symbolic formulae or explicit sets—in this case the sets will be over pairs of states. The symbolic formulae will be over distinct sets of variables—usually $\boxed{\text{VAR}}$ and VAR. For example, we would use the symbolic formula $\boxed{\mathsf{x}} > \mathsf{x}$ to represent the set of pair of states $\{(s,t) \mid s(\mathsf{x}) > t(\mathsf{x})\}$. As we did for sets, we define a semantic meaning, $[\![-]\!]$ (Note that the relational semantics $[\![-]\!]$ is subtely different than the semantics for states $[\![-]\!]$).

Definition 5 ($[\![-]\!]$) Assume that we are defining relations over states whose domain is the set of variables V. Assume that we have two mappings $\rho : X \to V$, and $\rho' : Y \to V$, where $X \cap Y = \emptyset$. Assume a formula f over variables X and Y. We define the relational meaning of f as

$$[\![f]\!] = \{(s,t) \mid f[(s \circ \rho) \cup (t \circ \rho')]\}$$

Unless explicitly noted otherwise, we will usually assume that $\rho(x) = \boxed{x}$ and $\rho'(x) = x$.

Example 2 Consider the formula $\boxed{\mathsf{x}} > \mathsf{x}$, which represents

$$\begin{array}{rcl} [\![\boxed{\mathsf{x}} > \mathsf{x}]\!] & = & \{(s,t) \mid (\boxed{\mathsf{x}} > \mathsf{x})[(s \circ \rho) \cup (t \circ \rho')]\} \\ & = & \{(s,t) \mid (\boxed{\mathsf{x}} > \mathsf{x})[(s \circ \boxed{-}) \cup t]\} \end{array}$$

The pair of states $(\{\mathsf{x} \mapsto 3, \mathsf{z} \mapsto 2\}, \{\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 2\})$ are in $[\![\boxed{\mathsf{x}} > \mathsf{x}]\!]$:

$$\begin{array}{rl} & (\boxed{\mathsf{x}} > \mathsf{x})[(\{\mathsf{x} \mapsto 3, \mathsf{z} \mapsto 2\} \circ \boxed{-}) \cup \{\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 2\}] \\ \Leftrightarrow & (\boxed{\mathsf{x}} > \mathsf{x})[(\{\boxed{\mathsf{x}} \mapsto 3, \boxed{\mathsf{z}} \mapsto 2\} \cup \{\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 2\}] \\ \Leftrightarrow & (\boxed{\mathsf{x}} > \mathsf{x})[(\{\boxed{\mathsf{x}} \mapsto 3, \boxed{\mathsf{z}} \mapsto 2, \mathsf{x} \mapsto 0, \mathsf{y} \mapsto 2\}] \\ \Leftrightarrow & 3 > 0 \\ \Leftrightarrow & \mathsf{true} \end{array}$$

$\star$

Observation 1 *For all formulae f, g over variables* VAR*,*

(i) $[\![\mathbf{true}]\!] = \mathcal{S}$

(ii) $[\![\mathbf{false}]\!] = \emptyset$
(iii) $[\![f \wedge g]\!] \Leftrightarrow [\![f]\!] \cap [\![g]\!]$
(iv) $[\![f \vee g]\!] \Leftrightarrow [\![f]\!] \cup [\![g]\!]$
(v) $[\![f \Rightarrow g]\!] \Leftrightarrow [\![f]\!] \subseteq [\![g]\!]$
(vi) $[\![\neg f]\!] = \neg[\![f]\!]$

Observation 2 *For all formulae f, g over variables V and $\boxed{V}$, and formulea A, B over variables V, $V' \cap (V \cup \boxed{V}) = \emptyset$,*

(i) $[\![\mathbf{true}]\!] = \mathcal{S} \times \mathcal{S}$
(ii) $[\![\mathbf{false}]\!] = \emptyset$
(iii) $[\![\forall V.\exists \boxed{V}.\ f]\!] = \overrightarrow{\lceil [\![f]\!]}$
(iv) $[\![\exists V.\ f]\!] = \overleftarrow{\lfloor [\![f]\!]}$
(v) $[\![f \wedge g]\!] \Leftrightarrow [\![f]\!] \cap [\![g]\!]$
(vi) $[\![f \vee g]\!] \Leftrightarrow [\![f]\!] \cup [\![g]\!]$
(vii) $[\![f \Rightarrow g]\!] \Leftrightarrow [\![f]\!] \subseteq [\![g]\!]$
(viii) $[\![\neg f]\!] = \neg[\![f]\!]$
(ix) $[\![\exists \boxed{V}.\boxed{X} \wedge R]\!] = [\![R]\!]([\![X]\!])$
(x) $[\![A]\!] \times [\![B]\!] = [\![\boxed{A} \wedge B]\!]$
(xi) $[\![R]\!]; [\![Q]\!] = [\![\exists V'.\ R[V/V'] \wedge Q[\boxed{V}/V']\ ]\!]$

From Observations 2(iv) and 2(iii), we will abuse notation and use $\overrightarrow{\lceil -}$ and $\overleftarrow{\lfloor -}$ on formulae as well as relations.

Due to Observations 1 and 2, we can use automatic decision procedures on formulae to establish properties about the sets and relations that they represent. For example, if we prove the validity of the formula $\forall \mathsf{x}, \boxed{\mathsf{x}}.\ \mathsf{x} = \boxed{\mathsf{x}} - 1 \Rightarrow \mathsf{x} < \boxed{\mathsf{x}}$ with an automatic decision procedure, then we know that $[\![\mathsf{x} = \boxed{\mathsf{x}} - 1]\!] \subseteq [\![\mathsf{x} < \boxed{\mathsf{x}}]\!]$.

1.4 Well-ordered sets and well-founded relations

In this section we describe what it means for a set to be *well ordered*, and a relation to be *well founded*.

Definition 6 (Total order) The structure $(S, \geq)$ forms a *total order* iff for all $a, b, c \in S$

- $a \geq a$ (reflexive),
- $a \leq b$ and $a \geq b$ then $a = b$ (antisymmetry),
- If $a \geq b$ and $b \geq c$ then $a \geq c$ (transitivity),
- $a \leq b$ or $a \geq b$ (totality),

Definition 7 (Well order) $(S, \geq)$ forms a *well order* iff it is a total order and every nonempty subset of S has a least element.

Example 3 The natural numbers, $\mathbb{N}$, are a well-ordered set, as in the worst case 0 is the least element of any subset. The integers, $\mathbb{Z}$, are not well ordered because there is no least element. However, for any integer constant $b \in \mathbb{Z}$, the set $\{x \mid x \in \mathbb{Z} \wedge x \geq b\}$ is a well-ordered set. $\star$

For convienance we will use the notation $\geq_b$ to represent the relation

$$x \geq_b y \triangleq x \geq y \wedge y \geq b$$

This allows us to create well orders from sets like the integers,

Example 4 The non-negative real numbers with relation $\geq$ are not a well-ordered set because there there is no least element in the open interval (0,1). The non-negative real numbers can be made into a well-ordered set when paired with an alternative comparison, for example: as $x \geq' y \triangleq x \geq y + 0.0001 \vee x = y$. $\star$

For convienance we will use the notation $\geq^b$ to represent the relation

$$x \geq^b y \triangleq x \geq y + b$$

This allows us to create well orders from sets like the positive reals.

Definition 8 (Sequences) We say that s is an S-sequence if $s = \langle s_1, s_2, \ldots \rangle$ and each $s_i \in S$. A finite sequence $s = \langle s_1, s_2, \ldots, s_n \rangle$ will have a last index $\mathsf{last}(s) = s_n$. Let $R \subseteq S \times S$. A finite sequence is said to be *permitted by* R iff $\forall i \in \{1, \ldots \mathsf{last}(s) - 1\}$. $(s_i, s_{i+1}) \in R$. An infinite sequence is permitted by R iff $\forall i.\ i > 0 \Rightarrow (s_i, s_{i+1}) \in R$. Let $I \subseteq S$. We say that s is permitted by a transition system $P = (I, R, S)$ (defined properly in Definition 12) if and only if s is permitted by R and $s_1 \in I$.

Definition 9 (Well-founded relations) A binary relation $R \subseteq S \times S$ is well-founded iff it does not permit infinite sequences.

Example 5 The $\boxed{\mathsf{x}} > \mathsf{x} \wedge \mathsf{x} > 0$ represents a well-founded relation if the variables range over the integers or natural numbers, but not if the variables range over the reals. The reason for well-foundedness in the integers or naturals is that, if we apply the relation point-wise to any sequence of naturals or integers, we'll see that the values along

the sequence are forced to go down towards (and eventually pass) a bound. Thus no permitted sequence can be infinite. In the reals the constraint $\boxed{\mathsf{x}} > \mathsf{x}$ does not require the value to go down enough to guarantee eventual progress to 0. The formula $\mathsf{x} \geq \boxed{\mathsf{x}} + 1 \wedge \mathsf{x} > 0$, on the other hand, represents a relation that is well founded in all three interpretations. $\star$

Theorem 1 *Assume that $(S, \geq)$ is a total order. $(S, \geq)$ is a well order iff the relation $x > y$ (defined as $x > y \triangleq x \geq y \wedge x \neq y$) is well founded on S-sequences.*

Proof

Well-ordered set $\Rightarrow$ Well-founded relation: By a contrapositive argument, assume that $>$ is not well founded, meaning in this case that there is an infinitely descending chain of S-elements. In this case there can be no least element. ✓

Well-ordered set $\Leftarrow$ Well-founded relation: Again, by a contrapositive argument. Assume that the infinite S-subset S' has no least element (the fact that every finite set has a least element can be established using the fact that S is a total order). Let $s_1 \in S'$. Since s_1 cannot be minimal we know that there exists an $s_2 \in S'$ such that $s_1 > s_2$, and an $s_3 \in S'$ such that $s_2 > s_3$, etc. Therefore, using the somewhat controversial axiom of dependent choice we can show that there exists an infinite sequence of S'-elements that is permitted by $>$ ✓

□

Observation 3 *If Q is well founded and $R \subseteq Q$, then R is well founded.*

Proof By contradiction. Assume that R is not well founded, but Q is. Therefore there exists an infinite sequence s such that $\forall i.(s_i, s_{i+1}) \in R$. Because $R \subseteq Q$, we know that $\forall i.(s_i, s_{i+1}) \in Q$, thus contradicting the claim that Q is well founded. □

Corollary 1 *If R is not well founded and $R \subseteq Q$, then Q is not well founded.*

Remark on Cantor's ordinal numbers. Cantor's ordinal numbersare often used in the literature discussing well-ordered sets. Cantor's ordinals are—in effect—a canonical representation for sets of well-ordered

sets who are all related in size. (*e.g.* the natural numbers and any isomorphic set can be represented by the ordinal number ω). In this book we avoid the ordinals for the reason that, although they can make a fundamental discussion more concise, they come at a great initial cost. Many distracting ideas and notation would need to be introduced.

1.5 Ranking functions and ranking relations

The traditional method of proving a relation $R \subseteq S \times S$ well founded is to find a map from the structure (R, S) to some known well-ordered set $(\geq, T)$ and then prove that the map is structure-preserving (*i.e.* that it is a homomorphism). Since we know that the relation $>$ (where $x > y \triangleq x \geq y \wedge x \neq y$) on T is well founded, by the properties of homomorphisms and Observation 3, we know that R too is well founded. Turing's maps are typically called *ranking functions*.

Definition 10 (Ranking function) A mapping f with a range to a well-ordered set is called a ranking function.

Definition 11 (Ranking relation) Let $f : X \to Y$ be a ranking function, where $(Y, \geq)$ is a well order. We say that $\lfloor \geq_f$ is f's *ranking relation*. In some cases we will want to use $\lfloor \geq_f$ to represent a formula, and in others to represent the relation explicitely. We will switch between these representations without mention unless the choice is not clear from the context.

Observation 4 *For any ranking function f to a well order $(Y, \geq)$, the relation $\lfloor \geq_f$ is well founded.*

Proof We know that there exists some Y such that $f : X \to Y$ such that $(\geq, Y)$ is a well-ordered set. Thus, due to Theorem 1, we know that $>$ is a well-founded relation on sequences drawn from Y. By way of contradiction, assume that $s_1, s_2, s_3, \ldots$ is an infinite sequence permitted by $\lfloor \geq_f$. This gives rise to the infinite sequence of Y-elements $f(s_1) > f(s_2) > f(s_3) > \ldots$. But this infinite sequence is not permitted, as $(\geq, Y)$ is well ordered. □

Example 6 Consider the example relation

$$R \triangleq \boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}} + 1$$

Assume that x and y range over the integers. To prove R well-founded we can use the ranking function $f(s) = s(\mathsf{x})$ and bound 0 to construct $\underline{\succeq_{0f}}$:

$$\begin{array}{rcl} \underline{\succeq_{0f}} & = & \{(s,t) \mid f(s) > f(t) \wedge f(s) \geq 0\} \\ & = & \{(s,t) \mid s(\mathsf{x}) > t(\mathsf{x}) \wedge s(\mathsf{x}) \geq 0\} \\ & = & [\![\boxed{\mathsf{x}} > \mathsf{x} \wedge \boxed{\mathsf{x}} \geq 0]\!] \end{array}$$

To prove that the inclusion $[\![R]\!] \subseteq \underline{\succeq_{0f}}$ holds we can use a decision procedure to prove the validity of the formula: $\forall \mathsf{x}, \mathsf{y}, \boxed{\mathsf{x}}, \boxed{\mathsf{y}}.\ R \Rightarrow \boxed{\mathsf{x}} > \mathsf{x} \wedge \boxed{\mathsf{x}} \geq 0$. By Observation 2, we know that $[\![R]\!] \subseteq \underline{\succeq_{0f}}$, thus proving $[\![R]\!]$ well founded. $\star$

1.6 Transition systems and supporting invariants

Definition 12 (Transition systems) We say that P is a *transition system* if $P = (I, R, S)$, where S is the (possibly infinite) set of program states represented as finite partial functions from Vars to Vals, $I \subseteq S$, and $R \subseteq S \times S$. We call I the *initial states*, and R the *update relation*.

Definition 13 (Reachable states) We call $R^*(I)$ the *reachable states* of the transition system $P = (I, R, S)$.

Definition 14 (Relational restrictions) We use the notation $R\downarrow_S$ to denote R when restricted to the cartesian product of S:

$$R\downarrow_S \triangleq R \cap (S \times S)$$

We define $R\downarrow_S$ to be R with the image restricted to S:

$$R\downarrow_S \triangleq R \cap (S \times \overrightarrow{[R]})$$

It should be clear that, for all R and S, $R\downarrow_S \subseteq R\downarrow_S$.

Definition 15 (Transition relation) Let $P = (I, R, S)$. We use the notation $R\downarrow_I$ to denote P's *transition relation*:

$$R\downarrow_I \triangleq R\downarrow_{R^*(I)}$$

The common wisdom when proving a relation well founded is that one must find both a ranking function *and a supporting invariant*. This is due to the fact that, in practice, relations are often only well founded

when restricted to the states reachable by the relation from some set of initial states—that is, a terminating transition system's update relation is often itself not well founded.

Observation 5 *For all R and I, $R\downarrow_I = R\downarrow_{R^*(I)} = R\downarrow_{R^*(I)}$.*

Proof $R\downarrow_I = R\downarrow_{R^*(I)}$ by definition. $R\downarrow_{R^*(I)} = R\downarrow_{R^*(I)}$ because $R(R^*(I)) \subseteq R^*(I)$. □

Definition 16 (Invariant) A set of states Q is an invariant of a transition system (R, I, S) iff $R^*(I) \subseteq Q$. An invariant is called an *inductive invariant* if it can be proved invariant by induction *i.e.*, showing that $I \subseteq Q$ and $R(Q) \subseteq Q$.

Note that, because $R\downarrow_I \subseteq R$, if R is well founded then $R\downarrow_I$ is also well founded. Clearly $R^*(I)$ is the strongest possible invariant, but it is not computable in theory and very difficult to compute in practice. Instead we usually look for a weaker (but easier to find) invariant Q that is strong enough to prove relations well founded. Because, by definition, $R^*(I) \subseteq Q$, then $R\downarrow_I \subseteq R\downarrow_Q \subseteq R\downarrow_Q$. Thus, if $R\downarrow_Q$ or $R\downarrow_Q$ are well founded, we know that $R\downarrow_I$ is well founded.

Example 7 Consider the relation $R \triangleq \boxed{\mathsf{x}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} + \boxed{\mathsf{y}} \wedge \mathsf{y} = \boxed{\mathsf{y}}$, where the variables range over the integers. R is not well founded if $\boxed{\mathsf{y}} \geq 0$. However, if we let the initial set of states be $I \triangleq \boxed{\mathsf{y}} \leq -1$, then $R\downarrow_I$ is well founded. To prove this we can let $Q = \mathsf{y} \leq -1$. Luckily, in this case, Q is an *inductive invariant*, meaning that we can show Q invariant simply via induction (*i.e.* $[\![I]\!] \subseteq [\![Q]\!]$ and $R(Q) \subseteq Q$) To prove that $[\![Q]\!] \times [\![Q]\!] \cap [\![\![R]\!]\!]$ is well founded we can show that $\boxed{Q} \wedge Q \wedge R \Rightarrow \underline{>_{0_\mathsf{x}}}$ (where the free variables are universilly quantified).

⋆

1.7 Composing well-founded relations

In many cases, constructing a ranking function for a complex relation can be a subtle art. As we have seen: *once we know* a ranking function, proving the necessary subset inclusion is usually not difficult—finding the ranking function argument is the hard part. This section describes a method for constructing arguments of well-foundedness via the composition of small sub-arguments. As we will see later, the method makes

the search for and construction of arguments easier, but makes checking the argument more difficult. Modern approaches are often based on this result and thus will be used heavily in subsequent chapters.

Theorem 2 *Let be a binary relation $R \subseteq S \times S$. Let $Q_1, Q_2, \ldots Q_n$ be a finite set of binary relations $Q_i \subseteq S \times S$ such that each Q_i is well founded. R is well founded iff $R^+ \subseteq Q_1 \cup Q_2 \cup \ldots \cup Q_n$.*

It is important to note that the union of well-founded relations is not necessarily well founded, thus making Theorem 2 a little surprising. To see why this is true consider the relations $P \triangleq 0 < \mathsf{x} \wedge \mathsf{x} < \boxed{\mathsf{x}}$ and $Q \triangleq 100 > \mathsf{x} \wedge \mathsf{x} > \boxed{\mathsf{x}}$. Both P and Q are well founded, but $P \cup Q$ is not. To see that $P \cup Q$ is not well founded consider the case where $s(\mathsf{x}) = 5$. In this case $(s, s) \in (P \cup Q)^2$, thus making $\{s \mid s(\mathsf{x}) = 5 \wedge s \in \mathcal{S}\}$ a valid recurrence set for $(P \cup Q)^2$. The use of transitive closure is the key to Theorem 2's soundness, as it rules out cases like this.

Example 8 Consider the relation

$$\begin{array}{rcl} R & \triangleq & (\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}}) \\ & \vee & (\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} \wedge \mathsf{y} = \boxed{\mathsf{y}} - 1) \end{array}$$

We can prove R well founded by showing $R \subseteq \underline{\geq 0_{\mathsf{x}+\mathsf{y}}}$. Alternatively we use Theorem 2 and establish well-foundeness via proof that $R^+ \subseteq \underline{\geq 0_{\mathsf{x}}} \cup \underline{\geq 0_{\mathsf{y}}}$. Note that we cannot prove the inclusion $R^+ \subseteq \underline{\geq 0_{\mathsf{x}}} \cup \underline{\geq 0_{\mathsf{y}}}$ directly with any known decision procedure, as they do not support transitive closure (transitive closure for infinite-state systems is undecidable in theory, and difficult in practice). In a later chapter we will show how techniques from program analysis can be adapted to address this class of question.

Note that finding $\underline{\geq 0_{\mathsf{x}}}$ and $\underline{\geq 0_{\mathsf{y}}}$ is, in a sense, easier than $\underline{\geq 0_{\mathsf{x}+\mathsf{y}}}$. The reason is that if we can often find the former argument by looking individually at R's disjuncts: $\underline{\geq 0_{\mathsf{x}}}$ is motivated by looking at the first disjunct in R (*i.e.* $\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}}$), and $\underline{\geq 0_{\mathsf{y}}}$ is motivated by the second.

$\star$

1.8 Proving well-foundeness by induction

Lemma 1 *Let be a binary relation R and Q be binary relatinos such that $R, Q \subseteq S \times S$. If $R \subseteq Q$ and $Q; R \subseteq Q$ then $R^+ \subseteq Q$.*

Example 9 Consider the relation from Example 8.

$$\begin{array}{rcl} R & \triangleq & (\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}}) \\ & \vee & (\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} \wedge \mathsf{y} = \boxed{\mathsf{y}} - 1) \end{array}$$

The previous suggestion was to use the argument of well-foundeness $R^+ \subseteq \lfloor \underline{>_0}_{\mathsf{x}} \cup \lfloor \underline{>_0}_{\mathsf{y}}$, but we gave no procedure for showing the inclusion with R^+. In order to use a decision procedure we can use the following inductive argument:

$$Q \triangleq (\lfloor \underline{>_0}_{\mathsf{x}} \cap \lfloor \underline{\geq}_{\mathsf{y}}) \cup (\lfloor \underline{>_0}_{\mathsf{y}} \cap \lfloor \underline{\geq}_{\mathsf{x}})$$

Note that Q is disjuctively well-founded, as $\lfloor \underline{>_0}_{\mathsf{x}}$ and $\lfloor \underline{>_0}_{\mathsf{y}}$ are both well-founded, and any subset of a well-founded relation is well-founded. We can then use a decision procedure to prove that $[\![R]\!] \subseteq [\![Q]\!]$ and $[\![Q; R]\!] \subseteq [\![Q]\!]$ (by showing that $R \Rightarrow Q$ and $\exists V'.\ Q[V/V'] \wedge R[\boxed{V}/V'] \Rightarrow Q$), thus (by Lemma 1) proving that $[\![R]\!]^+ \subseteq [\![Q]\!]$. As Q represents a disjunctively well-founded relation, by Theorem 2, we know that $[\![R]\!]$ is well founded. $\star$

1.9 Disproving well-foundedness

Until now we have considered only proving relations well founded, but not disproving—*i.e.* proving that a relation is not well-founded.

Definition 17 (Recurrence sets) Assume $P = (I, R, S)$. $Q \subseteq \mathcal{S}$ is a *recurrence set* of P if:

(i) $Q \subseteq \overleftarrow{R}\rfloor$
(ii) $Q \cap I \neq \emptyset$
(iii) $\forall x \in Q.\ \exists y.\ (x, y) \in R \wedge y \in Q$

Theorem 3 *The transition system P is non-terminating iff there exists a valid recurrence set Q for P*

Example 10 Consider the relation over $\mathcal{S}$:

$$R \triangleq \mathsf{x} > 0 \wedge (\mathsf{x} = \boxed{\mathsf{x}} - 1 \vee \mathsf{x} = \boxed{\mathsf{x}})$$

Let $I \triangleq \mathcal{S}$. The relation $R\downarrow_I$ is not well founded. To *prove* it not well founded (as opposed to simply failing to prove it well founded) we define $Q = \{s \mid s(\mathsf{x}) = 1 \wedge s \in \mathcal{S}\}$. $\overleftarrow{R}\rfloor = \{s \mid s(\mathsf{x}) > 0 \wedge s \in \mathcal{S}\}$, thus $Q \subseteq \overleftarrow{R}\rfloor$. Because $Q \subseteq I$ and $Q \neq \emptyset$, $Q \cap I \neq \emptyset$. Finally, $R(Q) = Q$, thus $\forall x \in Q. \exists y. (x, y) \in R \wedge y \in Q$.

⋆

1.10 Finite decomposition

Theorem 4 *Assume that* $v \in \text{VAR}$ *and that the set* L *is finite, where*

$$L = \{s(v) \mid s \in R^*(I)\}$$

$R\downarrow_I$ *is well founded if for all* $k \in L$, $(R\downarrow_I^+)\downarrow_{v=k}$ *is well founded.*

Proof By contradiction and the pigeon-hole principle. Assume that $s = s_1, s_2, s_3, \ldots$ is an infinite sequence such that $(s_1, s_2) \in R\downarrow_I$, $(s_2, s_3) \in R\downarrow_I$, etc. Because L is finite and $R^*(I)(v) = L$, we know that there exists a $c \in L$ such that $s_i(v) = c$ infinitely-often in s. Let s' be the infinite sequence of these states. We know that s' is in the sequences allowed by $R\downarrow_I^+ \cap \{(s,t) \mid s(v) = t(v) = c\}$ (*i.e.* $(R\downarrow_I^+)\downarrow_{v=c}$). But $(R\downarrow_I^+)\downarrow_{v=c}$ is well founded. □

Example 11 Consider the relation

$$\begin{array}{rcl} R & \triangleq & ((\mathsf{b} = 1 \wedge \boxed{\mathsf{b}} = 0) \vee (\mathsf{b} = 0 \wedge \boxed{\mathsf{b}} = 1)) \\ & \wedge & (\boxed{\mathsf{b}} = 1 \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \boxed{\mathsf{x}} > 0) \vee (\boxed{\mathsf{b}} = 0 \wedge \mathsf{x} = \boxed{\mathsf{x}}) \end{array}$$

In this case we could invent a fairly complex ranking function involving both x and b, or alternatively we can simply prove $R^+\downarrow_{\mathsf{b}=0} \subseteq \underline{\lfloor >_{0\mathsf{x}} \rfloor}$ and $R^+\downarrow_{\mathsf{b}=1} \subseteq \underline{\lfloor >_{0\mathsf{x}} \rfloor}$. Note that we can do slightly better—the following lemmas will allow us to eliminate one of these conjuncts. ⋆

Lemma 2 *Assume that* $v \in \text{VAR}$ *and that the set* $L = R^*(I)(v)$ *is finite. Let* k_1 *and* k_2 *be constants from* VAL. *Assume that, if* $(s,t) \in R$ *and* $t(v) = k_2$ *then* $s(v) = k_1$. $(R\downarrow_I^+)\downarrow_{v=l}$ *is well founded for each* $l \in L$ *iff* $(R\downarrow_I^+)\downarrow_{v=l}$ *is well founded for each* $l \in L - \{k_2\}$.

Proof By contradiction. Assume that there is an infinite sequence $s = s_1, s_2, s_3, \ldots$ allowed by R such that $s_i(v) = k_2$ infinitely often. By assumption, if $s_{i+1}(v) = k_2$ then $s_i(v) = k_1$. Thus $s_i(v) = k_1$ occurs infinitely often in s. But, by assumption, $(R\downarrow_I^+)\downarrow_{v=k_1}$ is well founded, meaning that s cannot be infinite and thus contradicting the starting assumption. □

$s \in \text{CmdSeqs}$	::=	finite sequence of c
$c \in \text{Cmds}$	::=	$v := t \mid v := \textbf{nondet} \mid \textbf{assume}(f)$
$f \in \text{Formulae}$	::=	$t_1 < t_2 \mid t_1 > t_2 \mid t_1 \geq t_2 \mid t_1 \leq t_2$
	$\mid$	$\neg f \mid f_1 \wedge f_2 \mid f_1 \vee f_2 \mid f_1 \Rightarrow f_2 \mid \textbf{true} \mid \textbf{false}$
$t \in \text{Terms}$	::=	$k \mid v \mid -t \mid t_1 + t_2 \mid t_1 - t_2 \mid t_1 \times t_2 \mid \ldots$
$k \in \mathbb{R}$		
$v \in \text{VAR}$		

Fig. 1.1. Terms, formulae, and program commands expressed in Backus-Naur form.

Lemma 3 *Assume that $v \in \text{VAR}$ and that the set $L = R^*(I)(v)$ is finite. Let k_1 and k_2 be constants from* VAL. *Assume that, if $(s,t) \in R$ and $s(v) = k_2$ then $t(v) = k_1$. $(R\downarrow_I^+)\restriction_{v=l}$ is well founded for each $l \in L$ iff $(R\downarrow_I^+\restriction_{v=l}$ is well founded for each $l \in L - \{k_2\}$.*

Proof By the same argument as Lemma 2 □

Lemmas 2 and 3 allow us to remove one of the checks from Example 11. We can now simply prove R well founded by proving $R^+\restriction_{\mathtt{b}=0} \subseteq \underline{\geq_{0\times}}$

1.11 Proving termination of programs

Until now we have considered methods of proving mathematical relations well founded. We now focus our attention on programs. In this section we describe a simple imperative programming language and discuss the semantic meaning that maps programs to the transition relations that they represent. We also provide some preliminary results which allow us to prove termination of programs.

Definition 18 (Terms, formulae, commands) Figure 1.1 gives the syntax for terms, formulae and commands. The semantics of formulae

is standard. Commands represent relations, as defined below:

$$\begin{array}{rcl} [\![x{:=}t]\!] & \triangleq & \{(s,s') \mid \forall v \in V - \{x\}.\ s(v) = s'(v) \wedge s'(x) = t[s]\} \\ [\![x{:=}\mathbf{nondet}]\!] & \triangleq & \{(s,s') \mid \forall v \in V - \{x\}.\ s(v) = s'(v)\} \\ [\![\mathbf{assume}(f)]\!] & \triangleq & \{(s,s') \mid \forall v \in V.\ s(v) = s'(v) \wedge s \in [\![f]\!]\} \end{array}$$

The meaning of a finite sequence of commands is the relational composition of the meaning of each command in the sequence.

Definition 19 (VarsOf) We define VarsOf(P) to be variables from Vars referenced in the commands of P.

Definition 20 (Programs) A program $\mathcal{P}$ is structurally represented as a rooted edge-labeled directed graph $(\mathcal{L}, \mathcal{E})$ where the nodes $\mathcal{L}$ represent set of locations and $\mathcal{E}$ represents the possible transitions between locations:

$$\mathcal{E} \subseteq \mathcal{L} \times \text{CmdSeqs} \times \mathcal{L}$$

We assume that the root is the location $\ell_{\mathbf{root}}$. The meaning of a program is a transition system, $(\mathcal{I}, \mathcal{R}) = [\![\mathcal{P}]\!]$, where $\exists p \notin \text{VarsOf}(\mathcal{P})$ such that

$$\begin{array}{rcll} \mathcal{R} \triangleq \{(s, s'') & \mid & \exists s', cs.\ (s,s') \in [\![cs]\!] \\ & \wedge & s'' = s'[p \mapsto \ell'] \\ & \wedge & s(p) \in \mathcal{L} \wedge s''(p) \in \mathcal{L} \\ & \wedge & (s(p), cs, s''(p)) \in \mathcal{E}. \\ & \} & \end{array}$$

$$\mathcal{I} \triangleq \{s \quad \mid \quad s(p) = \ell_{\mathbf{root}}\}$$

Example 12 Consider the following program text:

```
while x > 0 do
    x := x − 1;
od
```

In our internal representation we might represent this program using the following graph,: $\mathcal{L} = \{1, 2\}$, and

$$\mathcal{E} = \{(1, \langle \mathbf{assume}(\mathsf{x} > 0); \mathsf{x}{:=}\mathsf{x} - 1\rangle, 1), (1, \langle \mathbf{assume}(\neg(\mathsf{x} > 0))\rangle, 2)$$

In this case $\mathcal{R}$ equals the relation relation:

$$\begin{array}{lll}\{(s, s') & | & (s(\mathsf{pc}) = 1 \wedge s'(\mathsf{pc}) = 1 \wedge s(\mathsf{x}) > 0 \wedge s'(\mathsf{x}) = s(\mathsf{x}) - 1 \\ & & \quad \wedge \forall v \in \textsc{Vars} - \{\mathsf{pc}, \mathsf{x}\}.\ s(v) = s'(v)) \\ & \vee & (s(\mathsf{pc}) = 1 \wedge s'(\mathsf{pc}) = 2 \wedge s(\mathsf{x}) \leq 0 \\ & & \quad \wedge \forall v \in \textsc{Vars} - \{\mathsf{pc}\}.\ s(v) = s'(v)) \\ & \} & \end{array}$$

The transition system's set of initial states, $\mathcal{I}$, can be defined as the set $\mathcal{I} = \{s \mid s(\mathsf{pc}) = 1\}$. Note that we can use a decision procedure to prove that $\mathcal{R} \subseteq \underline{\succeq_{0_{\mathsf{x}-\mathsf{pc}}}}$. Note that in this case we are lucky, as the value of pc only goes up or stays the same.

$\star$

Definition 21 ($\mathcal{P}$-trace,$\mathcal{P}$-path) The (possibly finite) sequence s is a $\mathcal{P}$-trace if $s_0 \in \mathcal{I}$ and for all s indices i, $(s_{i-1}, s_i) \in \mathcal{R}$. Let $\pi(s) \triangleq s(\mathsf{pc})$. The function π, when applied to a sequence, returns a sequence in which π has been applied pointwise. A sequence p is a $\mathcal{P}$-path if there exists a $\mathcal{P}$-trace s such that $p = \pi(s)$. Let $\textsc{Paths}(\mathcal{P})$ be the set of all $\mathcal{P}$-paths, Let $\textsc{Traces}(\mathcal{P})$ be the set of all $\mathcal{P}$-traces. We define a $\mathcal{P}$-trace segment to be a finite sequence s such that $s_0 \in \mathcal{R}^*(\mathcal{I})$ and s indices i, $(s_{i-1}, s_i) \in \mathcal{R}$. Let $\textsc{TraceSegs}(\mathcal{P})$ be the set of all $\mathcal{P}$-trace segments.

Definition 22 (Cutpoints) $C \subseteq_{\text{fin}} \mathbb{N}$ is a valid *cutpoint-covering*of $\mathcal{P}$ if for any infinite $\mathcal{P}$-path s, there exists a $c \in C$ such that $s_i(\mathsf{pc}) = c$ for an infinite subset of s indices. Let $\textsc{Cutpoints}$ be some procedure that returns a valid set of cutpoints when passed a program, and let $\mathcal{C} \triangleq \textsc{Cutpoints}(\mathcal{P})$.

Observation 6 *Assume that v is the variable used to track program locations in the transition relation. $\mathcal{P}$ terminates if $\forall n \in \mathcal{C}.\ (\mathcal{R}\downarrow_{\mathcal{I}})\downarrow_{v=n}$ is well founded*

Proof By induction with Lemmas 2 and 3 on the bounded number of program locations not in $\mathcal{C}$. □

Note that, each obligation resulting from Observation 6 is, in essence, proving that the location ℓ cannot be visited infinitely-often during the program's execution. Thus, since no location can be proved infinitely-often, we know that the program terminates. When (in later chapters) we consider programs with nested loops, we will make improvements to

Observation 6 that allow us to ignore infinite executions through nested loops that also pass infinitely-often through an outer-loop.

1.12 Further reading

The reader interested in examining the original papers from which this chapter is drawn should begin with the proof of termination's undecidability [20, 19], and the seminal papers on proving program correctness (*e.g.* Turing's paper on proving programs correct [21], Floyd's paper on program semantics [10], Manna & Pnueli's books [14, 13]). Readers interested in well-ordered sets, well-founded relations, and the ordinals should refer to a text like [3]. Readers interested in disjunctive termination proofs (*i.e.* Theorem 2 and Lemma 1) should read Podelski & Rybalchenko's paper [16] together with Ramsey's original paper [17].

1.13 Exercises

(i) Assume that the variables in the following relations range over the integers. Which of the following relations are well founded? Which are not? Prove your answer by either finding (and proving the validity) of a ranking relation, or finding (and proving the validity) of a recurrence set. You are welcome to use an automatic decision procedure to discharge the obligations.

(a) $1 < 0$
(b) $0 < 1$
(c) $\mathsf{x} > \boxed{\mathsf{x}} \wedge \mathsf{x} < 1000$
(d) $\mathsf{x} > \boxed{\mathsf{x}} \wedge \mathsf{x} > 1000$
(e) $\mathsf{x} \geq \boxed{\mathsf{x}} + 1 \wedge \mathsf{x} < 1000$
(f) $\mathsf{x} \geq \boxed{\mathsf{x}} - 1 \wedge \mathsf{x} < 1000$
(g) $\mathsf{y} \geq \boxed{\mathsf{y}} + 1 \wedge \mathsf{z} = \boxed{\mathsf{z}} \wedge \boxed{\mathsf{z}} < 1000$
(h) $\mathsf{y} + 1 \geq \boxed{\mathsf{y}} \wedge \mathsf{z} = \boxed{\mathsf{z}} \wedge \boxed{\mathsf{z}} < 1000$
(i) $(\mathsf{x} = \boxed{\mathsf{x}} - 1 \vee \mathsf{x} = \boxed{\mathsf{x}} + 1) \wedge \mathsf{x} < 1000$
(j) $\mathsf{x} = \boxed{\mathsf{x}} - \boxed{\mathsf{z}} \wedge \boxed{\mathsf{x}} > 0$
(k) $\mathsf{x} = \boxed{\mathsf{x}} - \boxed{\mathsf{z}} \wedge \mathsf{x} > 0$
(l) $\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge (\boxed{\mathsf{x}} > 0 \vee \boxed{\mathsf{x}} < 200)$
(m) $\mathsf{x} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge [(\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \boxed{\mathsf{y}} = \boxed{\mathsf{y}}) \vee (\boxed{\mathsf{y}} = \boxed{\mathsf{y}} - 1 \wedge \mathsf{x} = \boxed{\mathsf{x}})]$
(n) $\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge [(\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}}) \wedge (\mathsf{y} = \boxed{\mathsf{y}} - 1 \wedge \mathsf{x} = \boxed{\mathsf{x}})]$
(o) $\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge [(\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}} + 1) \vee (\mathsf{y} = \boxed{\mathsf{y}} - 1 \wedge \mathsf{x} = \boxed{\mathsf{x}})]$

(p) $\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge [(\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}} + 1) \vee (\mathsf{y} = \boxed{\mathsf{y}} - 1 \wedge \mathsf{x} = \boxed{\mathsf{x}} + 1)]$

(q) $(\boxed{\mathsf{x}} > 0 \vee \boxed{\mathsf{y}} > 0) \wedge \mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}} - 1$

(r) $\boxed{\mathsf{x}} > 0 \wedge \mathsf{x} = \boxed{\mathsf{x}} - \boxed{\mathsf{y}} \wedge \mathsf{y} = \boxed{\mathsf{y}} + 1$

(ii) Reconsider the relations above in the case where the variables range over the real numbers? Which of the following relations are well founded? Which are not? Again, prove your answers.

(iii) Prove or disprove the following assertions:

(a) If R^2 is well founded, R is well founded.

(b) If R is well founded, R^2 is well founded.

(c) If R is well founded, R^+ is well founded.

(d) If R^+ is well founded, R is well founded.

(e) If R is well founded, $R \cap Q$ is well founded.

(f) If R is well founded, $R \cup Q$ is well founded.

(g) If R is well founded, R^{-1} is well founded.

(iv) Is the following relation well founded?

$$\boxed{\mathsf{x}} > 0 \wedge \boxed{\mathsf{y}} > 0 \wedge [(\mathsf{x} = \boxed{\mathsf{x}} + 1 \wedge \mathsf{y} = \boxed{\mathsf{y}} - 1) \vee (\mathsf{x} = \boxed{\mathsf{x}} - 1 \wedge \mathsf{y} = \boxed{\mathsf{y}})]$$

If so: Use Theorem 2 and Lemma 1 to prove the following relation well founded If not, find and prove the validity of a recurrence set.

(v) Translate the following program into the representation introduced in Section 1.11:

```
while x > 0 do
    x := x − 1;
    y := x;
    while y > 0 do
        y := y − 1;
    od
od
exit;
```

What is this program's semantic meaning (in the form of a relation)? Give a valid set of cutpoints for this program. Find and prove the validity of a ranking function that proves the relation well founded.

(vi) Use the techniques from Section 1.10 to prove the following relation well founded:

$$
\begin{array}{rl}
R \triangleq & \boxed{\mathsf{x}} = 0 \Rightarrow (\mathsf{x} = 1 \land \boxed{\mathsf{y}} > 0 \land \mathsf{y} = \boxed{\mathsf{y}}) \\
\land & \boxed{\mathsf{x}} = 1 \Rightarrow (\mathsf{x} = 2 \land \mathsf{y} = \boxed{\mathsf{y}}) \\
\land & \boxed{\mathsf{x}} = 2 \Rightarrow (\mathsf{x} = 3 \land \mathsf{y} = \boxed{\mathsf{y}} - 1) \\
\land & \boxed{\mathsf{x}} = 3 \Rightarrow (\mathsf{x} = 0 \land \mathsf{y} = \boxed{\mathsf{y}}) \\
\land & (\boxed{\mathsf{x}} = 3 \lor \boxed{\mathsf{x}} = 2 \lor \boxed{\mathsf{x}} = 1 \lor \boxed{\mathsf{x}} = 1 \lor \boxed{\mathsf{x}} = 0)
\end{array}
$$

Bibliography

J. Berdine, A. Chawdhary, B. Cook, D. Distefano, and P. O'Hearn. Variance analyses from invariance analyses. In *POPL'07: Programming Language Design and Implementation*, 2007.

J. Berdine, B. Cook, D. Distefano, and P. O'Hearn. Automatic termination proofs for programs with shape-shifting heaps. In *CAV'06: Computer Aided Verification*, 2006.

G. Cantor. *Contributions to the Founding of the Theory of Transfinite Numbers*. Dover, 1955.

A. Chawdhary, B. Cook, S. Gulwani, M. Sagiv, and H. Yang. Ranking abstractions. In *ESOP'08: European Symposium on Programming*, 2008.

B. Cook. The foundations of program termination. In *Lecture Notes of the Marktoberdorf 2008 Summer School (to appear)*, 2008.

B. Cook, A. Gotsman, A. Podelski, A. Rybalchenko, and M. Vardi. Proving that programs eventually do something good. In *POPL'07: Programming Language Design and Implementation*, 2007.

B. Cook, S. Gulwani, T. Lev-Ami, A. Rybalchenko, and M. Sagiv. Proving conditional termination. In *CAV'08: Computer Aided Verification*, 2008.

B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *PLDI'06: Programming Language Design and Implementation*, 2006.

B. Cook, A. Podelski, and A. Rybalchenko. Proving thread termination. In *PLDI'07: Programming Language Design and Implementation*, 2007.

R. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, 1967.

A. Gupta, T. Henzinger, R. Majumdar, A. Rybalchenko, and R. Xu. Proving non-termination. In *POPL'08: Principles of Programming Languages*, 2008.

S. Magill, J. Berdine, E. Clarke, and B. Cook. Arithmetic strengthening for shape analysis. In *SAS'07: Static Analysis Symposium*, 2007.

Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Progress*. Springer, 1995.

Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Safety*. Springer, 1995.

A. Podelski and A. Rybalchenko. A complete method for the synthesis of

linear ranking functions. In *VMCAI'04: Verification, Model Checking, and Abstract Interpretation*, pages 239–251, 2004.

A. Podelski and A. Rybalchenko. Transition invariants. In *LICS'04: Logic in Computer Science*, pages 32–41. IEEE, 2004.

F. Ramsey. On a problem of formal logic. *London Math. Soc.*, 30:264–286, 1930.

G. Stix. Send in the Terminator. *Scientific American Magazine*, November 2006.

C. Strachey. An impossible program. *Computer Journal*, 7(4):313, 1965.

A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *London Mathematical Society*, 42(2):230–265, 1936.

A. Turing. Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating machines*, pages 67–69, 1949.