Formalisms Every Computer Scientist Should Know

License: This work is marked with CC0 1.0. To view a copy of this license, visit
http://creativecommons.org/publicdomain/zero/1.0

Contents

Class 1

0.1 Tools

- Lean or Coq
- CVC5 or Z3
- possibly a model checker
- professional version of ChatGPT

0.2 Syllabus

1. MATH ("Informalism")
    - proofs (natural deduction)
    - fixpoints (induction, coinduction)
2. DECLARATIVE LOGIC
    - syntax (rules) vs. semantics (models)
    - propositional, predicate, modal logic
    - decision procedures (SAT, SMT)
3. FUNCTIONS
    - λ calculus, typed λ
    - SOS (structured operational semantics), rewriting
    - "propositions-as-types" (connection to logic)
4. PROCESSES (CONCURRENT)*
    - CCS, Petri nets
    - (bi-) simulation
5. CIRCUITS*
    - boolean, sequential, dataflow (Kahn nets)
    - interfaces

6. STATE TRANSITION SYSTEMS*
   - (ω-) automata, games, timed, probabilistic, pushdown
   - programs, Turing machines
   - grammars (Chomsky hierarchy)
7. DECLARATIVE SPECIFICATION
   - Hoare logics, separation logic
   - temporal logics (LTL, CTL, ATL)
   - partial correctness vs. termination, safety vs. liveness

*: Operational.

0.3 Math

Definition 1. *A real b is a bound of a function f from $\mathbb{R}$ to $\mathbb{R}$ if for all x in $\mathbb{R}$, we have $f(x) \leq b$.*

Definition 2. *Given two functions f and g from $\mathbb{R}$ to $\mathbb{R}$, their sum is the function $f+g$ such that for all x in $\mathbb{R}$, we have $(f+g)(x) = f(x)+g(x)$.*

Theorem 1. *For all functions f and g from $\mathbb{R}$ to $\mathbb{R}$, if f and g are bounded, then $f+g$ is bounded.*

Proof.
1. Consider arbitrary functions $\hat{f}$ and $\hat{g}$ from $\mathbb{R}$ to $\mathbb{R}$.
2. Assume $\hat{f}$ and $\hat{g}$ are bounded.
3. Show that $\hat{f}+\hat{g}$ is bounded.
4. ($2 \rightarrow$) Let $\hat{a}$ be a bound for $\hat{f}$, and $\hat{b}$ be a bound for $\hat{g}$.
5. We show that $\hat{a}+\hat{b}$ is a bound for $\hat{f}+\hat{g}$.
6. Consider an arbitrary real $\hat{x}$.
7. Show $(\hat{f}+\hat{g})(x) \leq \hat{a}+\hat{b}$.
8. (Definition of sum) $(\hat{f}+\hat{g})(\hat{x}) = \hat{f}(\hat{x})+\hat{g}(\hat{x})$.
9. (Definition of bound) $\hat{f}(\hat{x}) \leq \hat{a}$ and $\hat{g}(\hat{x}) \leq \hat{b}$.
10. The rest follows from "arithmetic".

□

Homework. Prove the Schröder-Bernstein theorem "in this style".

Definition 3. *Two sets A and B are equipollent ("have the same size") if there is a bijection from A to B.*

Definition 4. *A function f from A to B is*

1. *one-to-one if for all x and y in A, if $x \neq y$, then $f(x) \neq f(y)$.*
2. *onto if for all z in B, there exists x in A such that $f(x) = z$.*

3. *bijective if f is one-to-one and onto.*

Goals	Knowledge	Outermost symbol
Show for all x, $G(x)$. Consider arbitrary $\hat{x}$. Show $G(\hat{x})$.	We know for all x, $K(x)$. In particular, we know $K(\hat{t})$. $\hat{t}$: term containing only constants.	$\forall$
Show there exists x s.t. $G(x)$. We show that $G(\hat{t})$. $\hat{t}$: term containing only constants.	We know there exists x s.t. $K(x)$. Let $\hat{x}$ be s.t. $K(\hat{x})$.	$\exists$

Class 2

Goal	Knowledge	Outermost symbol
Show for all x, $G(x)$. Consider arbitrary $\hat{x}$. Show $G(\hat{x})$	We know for all x, $K(x)$ In particular we know $K(\hat{t})$ for constant $\hat{t}$	$\forall$
Show: exists x s.t. $G(x)$. We show $G(\hat{t})$	We know exists x s.t. $K(x)$ Let $\hat{x}$ be s.t. $K(x)$	$\exists$
Show G_1 iff G_2 1. Show if G_1 then G_2 2. Show if G_2 then G_1	We know K_1 iff K_2 In particular we know if K_1 then K_2 and if K_2 then K_1	$\iff$
Show if G_1 then G_2 Assume G_1 Show G_2	We know if K_1 then K_2 1. To show K_2 it suffices to show K_2 2. Know K_1, Also know K_2	$\Rightarrow$
Show G_1 and G_2 1. Show G_1 2. Show G_2	Know K_1 and K_2 1. Also Know K_1 2. Also Know K_2	$\wedge$
Show G_1 or G_2 1. Assume $\neg G_1$, show G_2 2. Assume $\neg G_2$, show G_1	We know K_1 or K_2. Show G. 1. Assume K_1, Show G 2. Assume K_2, Show G Case split $\uparrow$	$\vee$
Move Negation Inside, as far as possible		$\neg$

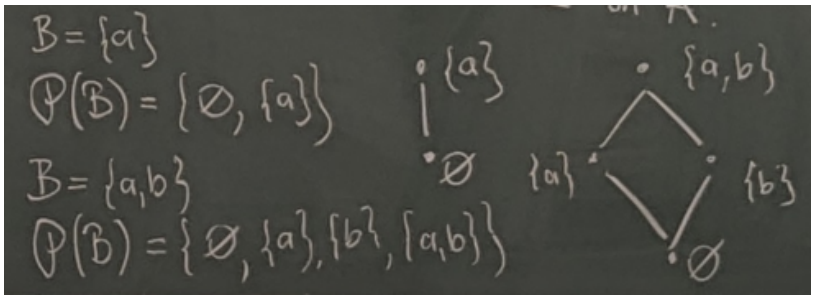
0.4 Lattices and Fixpoints

We begin by defining relations and their properties.

Definition 5. *A binary **relation** R on a set A is a subset $R \subset A \times A$.*
*The relation R is **reflexive** if for all x in A, we have $R(x,x)$.*
*The relation R is **Antisymmetric** if for all x and y in A, if $R(x,y)$ and $R(y,x)$ then $x = y$.*
*The relation R is **transitive** if for all x, y and z in A, if $R(x,y)$ and $R(y,z)$ then $R(x,z)$.*
*The relation R is a **partial order** if R is reflexive, antisymmetric and transitive.*
*A **Poset** $(A, \sqsubseteq)$ is a set A and a partial order $\sqsubseteq$ on A.*

Example 1. *The pair $(\mathbb{N}, \leq)$ where $\mathbb{N}$ is the set of natural numbers, is a poset.*
For every set B, we have $(\mathscr{P}(B), \subseteq)$ where $\mathscr{P}(B)$ is the powerset of B, is a poset.

Definition 6.
- *Let $(A, \sqsubseteq)$ be a poset. A function F from A to A is **monotone** (order-preserving, homomorphism) if for all x and y in A, if $x \sqsubseteq y$, then $F(x) \sqsubseteq F(y)$.*
- *F has a fixpoint x in A if there exists x in A such that $F(x) = x$.*



- *x in A is a pre-fixpoint of F if $x \sqsubseteq F(x)$ and is a post-fixpoint of F, if $F(x) \sqsubseteq x$.*

Definition 7. *Let $(A, \sqsubseteq)$ be a poset.*

- *x in A is an **upper bound**(lower bound) on a subset B of A if for all y in B, it holds that $y \sqsubseteq x$ ($x \sqsubseteq y$).*
- *x is the least upper bound of B if (i) x is an upper bound of B and (ii) for all upper bounds y of B, we have $x \sqsubseteq y$. We denote such x by $\bigsqcup B$.*
- *x is the greatest lower bound of B if (i) x is a lower bound of B and (ii) for all lower bounds y of B, we have $y \sqsubseteq x$. We denote such x by $\bigsqcap B$.*

Example 2.
- *Consider the poset $(\mathbb{N}, \leq)$. Then for any $B \subseteq \mathbb{N}$, if B is finite, $\bigsqcup B$ is well-defined and equal to $\max B$. If B is infinite, then $\bigsqcup B$ does not exist.*
- *Consider the poset $(\mathbb{N} \cup \{\infty\}, \leq)$ where for all x in $\mathbb{N}$, it holds that $x \leq \infty$. Then for all $B \subseteq \mathbb{N}$, the least upper bound $\bigsqcup B$ is well-define.*
- *Let A be any set and consider the poset $(\mathscr{P}(A), \subseteq)$. For any subset B of $\mathscr{P}(A)$, it holds that $\bigsqcup B = \bigcup B$ and $\bigsqcap B = \bigcap B$.*

Definition 8. *Poset $(A, \sqsubseteq)$ is a **complete-lattice** if for all $B \subseteq A$, both $\bigsqcap B$ and $\bigsqcup B$ exist.*

Example 3. *Let $(A, \sqsubseteq)$ be a complete-lattice.*

- $\bigsqcup A = \top$
- $\bigsqcap A = \bot$
- $\bigsqcup \varnothing = \bot$
- $\bigsqcap \varnothing = \top$

Theorem 2 (Knaster-Tarski)**.** *For every complete lattice $(A, \sqsubseteq)$ and monotone function F on A, it holds that*

1. *$\bigsqcup \{x \in A | x \sqsubseteq F(x)\}$ is the unique greatest fixpoint of F.*
2. *$\bigsqcap \{x \in A | F(x) \sqsubseteq x\}$ is the unique least fixpoint of F.*

Homework 1. *Prove the Knaster Tarski Theorem.*

Class 3

Definition 9 (Prefixpoint)**.** *Consider a lattice $(A, \sqsubseteq)$ and a function $f : A \to A$. The set of prefixes is*

$$\{x \in A : x \sqsubseteq f(x)\}.$$

Definition 10 (Postfixpoint)**.** *Consider a lattice $(A, \sqsubseteq)$ and a function $f : A \to A$. The set of postfixes is*

$$\{x \in A : f(x) \sqsubseteq x\}.$$

Definition 11 (gfp and lfp)**.** *Consider a complete lattice $(A, \sqsubseteq)$ and a function $f : A \to A$. Then,*

$$\begin{aligned} \text{gfp} f &\coloneqq \bigsqcup \{x \in A : x \sqsubseteq f(x)\} \\ \text{lfp} f &\coloneqq \bigsqcap \{x \in A : f(x) \sqsubseteq x\}. \end{aligned}$$

Theorem 3 (Fixpoints)**.** *Consider a complete lattice $(A, \sqsubseteq)$ and a monotonic function $f : A \to A$. Then,* gfp$f$ *and* lfp$f$ *are fixpoints of f and, for all fixpoints x of f, we have* lfp$f \sqsubseteq x \sqsubseteq$ gfp$f$.

Definition 12 ($\bigsqcup$-continuous)**.** *Consider a complete lattice $(A, \sqsubseteq)$. A function $f : A \to A$ is $\bigsqcup$-continuous if, for all increasing sequences $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq x_2 \sqsubseteq \ldots$, we have*

$$f\left(\bigsqcup \{x_n : n \in \mathbb{N}\}\right) = \bigsqcup \{f(x_n) : n \in \mathbb{N}\}.$$

Definition 13 ($\bigsqcap$-continuous)**.** *Consider a complete lattice $(A, \sqsubseteq)$. A function $f : A \to A$ is $\bigsqcap$-continuous if, for all increasing sequences $x_0 \sqsupseteq x_1 \sqsupseteq x_2 \sqsupseteq x_2 \sqsupseteq \ldots$, we have*

$$f\left(\bigsqcap \{x_n : n \in \mathbb{N}\}\right) = \bigsqcap \{f(x_n) : n \in \mathbb{N}\}.$$

Lemma 1. *$\bigsqcup$-continuous implies monotonicity and $\bigsqcap$-continuous implies monotonicity.*

Theorem 4 (Constructive fixpoints)**.** *Consider a complete lattice $(A, \sqsubseteq)$ and a monotonic function $f : A \to A$. Then,*

$$\begin{aligned} \text{lfp} f &= \bigsqcup \{f^n(\bot) : n \in \mathbb{N}\} \\ \text{gfp} f &= \bigsqcap \{f^n(\top) : n \in \mathbb{N}\}. \end{aligned}$$

Homework 2. *Prove this theorem.*

Definition 14 ($\mathbb{N}$)**.** *Define $\mathbb{N}$ as the smallest set X such that*

1. $0 \in X$

2. *if $n \in X$, then $Sn \in X$*

In the definition of $\mathbb{N}$, we consider a universal set U sufficiently big, the complete lattice $(2^U, \subseteq)$ and the function on sets given by $f(Y) \coloneqq \{0\} \cup \{Sn : n \in Y\}$. Then, $\mathrm{lfp} f = \mathbb{N}$.

Definition 15 (Set of words)**.** *Consider a finite alphabet Σ. Define Σ^* as the smallest set X such that*

1. $\varepsilon \in X$
2. *for all $a \in \Sigma$, we have $aX \subseteq X$.*

A few remarks are in place.

- Inductively defined sets are countable and consist of finite elements.
- Inductively defined sets can be written as rules $x \Rightarrow f(x)$ meaning that, if $x \in X$, then $f(x) \in X$.
- Inductively defined sets allow proof by induction. Consider proving that for all $x \in X$ we have $G(x)$. This can be proven by showing
  1. $G(\bot)$
  2. For all $x \in X$, if $G(x)$, then $G(f(x))$

Definition 16 (Balanced binary sequences)**.** *Define the set S as the largest set X such that*

1. $X \subseteq 01X \cup 10X$.

In the definition of balanced binary sequences, we consider the complete lattice $(\Sigma^\omega, \subseteq)$ and the function on sets given by $f(X) \coloneqq 01X \cup 10X$. Then, balanced binary sequences correspond to $\mathrm{gfp} f$.

Definition 17 (Interval $[0,1]$)**.** *Define the set S as the largest set X such that*

1. $X \subseteq 0X \cup 1X \cup \ldots \cup 9X$.

A few remarks are in place.

- Coinductively defined sets are uncountable and consist of infinite elements.
- Coinductively defined sets can be written as rules $x \Leftarrow f(x)$ meaning that, for all $y \in X$, there exists x such that $y = f(x)$ and $x \in X$.
- Coinductively defined sets allow proof by coinduction. Consider proving that for all x, if $G(x)$, then $x \in X$. This can be proven by showing
  1. For all x and i, if $G(f_i(x))$, then $G(x)$,

  where $\{f_1, \ldots, f_n\}$ is the set of rules that define the set X.

Homework 3 (Prove balanced binary sequences)**.** *Consider S generated by the rules $X \Leftarrow 01X$ and $X \Leftarrow 10X$. Prove that, for all binary words x, we have that x in$S$ if and only if every finite prefix of even length of x has the same number of* 0*s and* 1*s.*

Hints.

1. The direction $\Leftarrow$ can be proven by coinduction.
2. The direction $\Rightarrow$ can be proven by induction on the length of the prefix.

Class 4

Definition 18 (Merge)**.** *Let Σ^* be an alphabet with linear operator $\leq$. Then for all $x, y \in \Sigma^*$ and $a, b \in \Sigma$ we have that:*

- $merge(\varepsilon, x) = x$,
- $merge(y, \varepsilon) = y$, *and*
- $merge(a \cdot x, b \cdot y) = a \cdot merge(x, b \cdot y)$ *if* $a \leq b$, *otherwise* $b \cdot merge(a \cdot x, y)$.

Definition 19. *For all $y \in \Sigma^*$, we have that $\varepsilon, y = y$.*

Definition 20. *For all letters $a \in \Sigma$, and strings $x, y \in \Sigma^*$, we have that $(a \cdot x), y = a \cdot (x, y)$.*

Theorem 5. *$\forall x, y, z \in \Sigma^*$ we have that $(x, y), z = x, (y, z)$.*

Proof. Consider arbitrary $\hat{y}, \hat{z} \in \Sigma^*$. The goal is to show that for all $x \in \Sigma^*$, holds that $(x, \hat{y}), \hat{z} = x, (\hat{y}, \hat{z})$. We use induction on x and we have the following cases:

- $x = \varepsilon$. We get that $(\varepsilon, \hat{y}), \hat{z} = \varepsilon, (\hat{y}, \hat{z})$. Using Definition 19 we get $\hat{y}, \hat{z} = \hat{y}, \hat{z}$.
- $x = \hat{a} \cdot \hat{u}$ for some $a \in \Sigma$ and $\hat{u} \in \Sigma^*$, we have that the induction hypothesis is $(\hat{u}, \hat{y}), \hat{z} = \hat{u}, (\hat{y}, \hat{z})$ and the goal is to show that
$$(\hat{a} \cdot \hat{u}, \hat{y}), \hat{z} = (\hat{a} \cdot \hat{u}), (\hat{y}, \hat{z}). \tag{1}$$
If we apply Definition 20 we get
$$(\hat{a} \cdot (\hat{u}, \hat{y})), z = \hat{a} \cdot (\hat{u}, (\hat{y}, \hat{z})). \tag{2}$$
Applying Definition 20 to the left side and the induction hypothesis to the right side of equation 2 we get
$$\hat{a} \cdot ((\hat{u}, \hat{y}), \hat{z}) = \hat{a} \cdot ((\hat{u}, \hat{y}), \hat{z}). \tag{3}$$

□

Definition 21. *$reverse(\varepsilon) = \varepsilon$.*

Definition 22. *For all $a \in \Sigma$ and $x \in \Sigma^*$, we have that $reverse(a \cdot x) = reverse(x) \cdot a$.*

Definition 23. *For all $y \in \Sigma^*$, we have that $r(\varepsilon, y) = y$.*

Definition 24. *For all $a \in \Sigma$ and $x, y \in \Sigma^*$, we have that $r(a \cdot x, y) = r(x, a \cdot y)$.*

Theorem 6. *For all $x \in \Sigma^*$, $reverse(x) = r(x, \varepsilon)$.*

Proof. We prove it by induction on x.

- For the base case we have to prove that for $x = \varepsilon$, $reverse(\varepsilon) = r(\varepsilon, \varepsilon)$. Using Definitions 21 and 23 we get that $\varepsilon = \varepsilon$.
- As induction hypothesis we assume that for all $y \in \Sigma^*$ and $x = \hat{a} \cdot \hat{u}$, where $\hat{a} \in \Sigma$ and $\hat{u} \in \Sigma^*$, it holds that $reverse(\hat{u}), y = r(\hat{u}, y)$. Then we have to show that

$$reverse(\hat{a} \cdot \hat{u}), y = r(\hat{a} \cdot \hat{u}, y). \tag{4}$$

  - Consider an arbitrary $\hat{y}$. If we apply Definitions 22 and 24 to the left and the right side respectively we get

$$(reverse(\hat{u}), \hat{a}), \hat{y} = r(\hat{u}, \hat{a} \cdot \hat{y}). \tag{5}$$

  - Then we apply Theorem 5 to the left side and get

$$reverse(\hat{u}), (\hat{a}, \hat{y}) = r(\hat{u}, \hat{a} \cdot \hat{y}). \tag{6}$$

    Finally, we apply the induction hypothesis to the left side of our previous equation and get

$$r(\hat{u}, \hat{a} \cdot \hat{y}) = r(\hat{u}, \hat{a} \cdot \hat{y}). \tag{7}$$

□

Definition 25 (Odd and Even)**.** *For all $x \in \Sigma^*$ and $a \in \Sigma$, we have that $odd(\varepsilon) = \varepsilon$ otherwise $odd(a \cdot x) = a \cdot even(x)$, where $even(\varepsilon) = \varepsilon$ and $even(a \cdot x) = odd(x)$.*

Definition 26 (Well-founded)**.** *A binary relation $\prec$ on a set A is well-founded if there is no infinite descending sequence $x_0 \succ x_1 \succ x_2 \succ \ldots$ on A.*

Example 4. *Two examples are $<$ over the set of natural numbers and "shorter than" on Σ^*.*

Well-founded induction principle (for well-founded $\prec$) . In order to show $\forall x \in A$, $G(x)$:

1. Consider an arbitrary $\hat{x} \in A$.
2. Assume $\forall y \prec \hat{x}$, $G(y)$.
3. Show $G(\hat{x})$.

Definition 27 (Mergesort)**.** *For all $x \in \Sigma^*$, we have that*

$$mergesort = merge(mergesort(odd(x)), mergesort(even(x))). \tag{8}$$

Definition 28 (Sorted)**.** *For all $a \in \Sigma$, we have that $sorted(\varepsilon)$, $sorted(a \cdot \varepsilon)$ and for all $x \in \Sigma^*$, we have that $sorted(a \cdot b \cdot x)$ iff $a \leq b$ and $sorted(b \cdot x)$.*

Lemma 2 (Homework)**.** *For all $x, y \in \Sigma^*$, if $sorted(x)$ and $sorted(y)$, then $sorted(merge(x, y))$.*

Theorem 7 (Homework)**.** *For all $x \in \Sigma^*$, we have that $sorted(mergesort(x))$.*

Definition 29 (Homework)**.** *Write a definition of permutation.*

Theorem 8 (Homework)**.** *For all $x \in \Sigma^*$, we have that $permutation(x, mergesort(x))$.*

0.4.1 Humans and Monkeys

Definition 30 (D1)**.** *for all x and y, we have $x > y$ iff parent(x,y) or there exists z such that parent(x,z) and $z > y$.*

Definition 31 (D2)**.** *For all x and y, we have $x > y$ iff parent(x,y) or there exists z such that $x > z$ and parent(z,y).*

Axiom 0.4.1. *$<$ is well-founded.*

Axiom 0.4.2. *For all x, we have that $h(x)$ implies $\neg m(x)$ and $m(x)$ implies $\neg h(x)$.*

Theorem 9 (Homework)**.** *Provide a proof of the following using D1 and then using D2. If there exist x and y such that $x > y$ and m(x) and h(y), then there exist x and y such that parent(x,y) and m(x) and h(y).*

Class 6

Formal system F is a set of rules. Rule is a finite set of (formulas) premises $p_0, \ldots, p_k$ and (a formula called) conclusion c. We usually have infinitely many rules but only finitely many different rule schemata. For example, schema $\phi \to \phi$ gives infinitely many rules like $p_3 \to p_3$. Axiom is a rule without premises.

Proof (derivation) is a finite sequence of formulas $\phi_0, \ldots, \phi_n$ such that every formula in the sequence is

- either an axiom (which can be viewed as a special case of the following);
- or the conclusion of a rule whose premises occur earlier in the sequence.

This is a linear view.

Linear view is usually easier for proving meta theorems. Tree view (inductive definition) is usually better in practice.

Theorem is a formula that occurs in a proof. We distinguish the following:

- $\vdash \phi$ ... "ϕ is a theorem (of the formal system F)" (has a proof) [syntax]
- $\vDash \phi$ ... "ϕ is valid (ϕ is tautology)" (is true in all models) [semantics]

Formal system equipped with semantics is called a logic. Most of logic is about establishing $\vdash \phi$ iff $\vDash \phi$.

Rule R is sound iff [if all premises of R are valid, then the conclusion of R is valid]. Formal system F is sound iff all rules are sound (or equivalently, every theorem is valid). Formal system F is complete iff every valid formula is a theorem. Formal system F is consistent unless $\vdash \bot$ (or equivalently, there exists a formula that is not a theorem). Rule R is derivable in F iff [for all formulas ϕ, $\underset{F \cup \{R\}}{\vdash} \phi$ iff $\underset{F}{\vdash} \phi$]. Rule R is admissible in F iff $F \cup \{R\}$ is still consistent. Formula ϕ is expressible in a logic L iff [there exists a formula ψ of L such that, for all interpretations v, $[[\phi]]_v = [[\psi]]_v$. For example $\phi_1 \wedge \phi_2$ is expressible using only $\neg$ and $\vee$ (de Morgan) as $\psi = \neg(\neg\phi_1 \wedge \neg\phi_2)$.

We can enumerate all theorems by systematically enumerating all possible proofs. The proof is a witness for validity. Sound formal system gives a sound procedure for validity (but not necessarily complete). Sound complete formal system gives a sound semi-complete procedure for validity (may not terminate on inputs that represent a formula that is not valid). To get a decision procedure (sound and complete procedure for validity), we need both (1) sound complete formal system for validity, and (2) sound complete formal system for satisfiability (to define a formal system for satisfiability, replace "formulas" (ϕ is valid) by "judgements" (ϕ is satisfiable); all axioms are satisfiable, all rules go from satisfiables to satisfiable). For every input ϕ, one of them will eventually terminate. Conclude; either ϕ is valid, or $\neg\phi$ is satisfiable (which means that ϕ is not valid). Recall that, if both a set and its complement are recursively-enumerable, the set is recursive (decidable).

Example (formal system for unsatisfiability):

$$\frac{\Gamma[\bot] \qquad \Gamma[\top]}{\Gamma[p]}$$

0.5 Hilbert formal system for propositional logic

Hilbert system uses connectives $\to$ and $\neg$ only. Hilbert system has three axioms and one rule – modus ponens (MP):

$$\frac{\phi \qquad\qquad \phi \to \psi}{\psi}$$

Axioms:

- (K): $\phi \to \psi \to \phi$
- (S): $(\phi \to \psi \to \chi) \to ((\phi \to \psi) \to (\phi \to \chi))$
- (em): $(\neg\phi \to \neg\psi) \to (\psi \to \phi)$

Example (prove $\phi \to \phi$ in Hilbert system):
(K) $\phi \to (\psi \to \phi) \to \phi$
(S) $(\phi \to (\psi \to \phi) \to \phi) \to ((\phi \to \psi \to \phi) \to (\phi \to \phi))$
(MP) $(\phi \to \psi \to \phi) \to (\phi \to \phi)$
(K) $\phi \to \psi \to \phi$
(MP) $\phi \to \phi$

Notation: $\Gamma \vdash \phi$ means $\underset{F \cup \Gamma}{\vdash} \phi$ (the set of formulas Γ is used as added axioms)
Metatheorem ("deduction theorem"): $\Gamma \vdash \phi \to \psi$ iff $\Gamma, \phi \vdash \psi$
Metaproof:
"$\Longrightarrow$": One application of modus ponens.
"$\Longleftarrow$": Assume ψ has a proof π using axioms Γ, ϕ, (K), (S), (em). Show that $\phi \to \psi$ has a proof π' using Γ, (K), (S), (em) — induction on length n of π.
Case $n = 1$: ψ must be an axiom. Either $\psi \in \Gamma \cup \{\mathrm{K}, \mathrm{S}, \mathrm{em}\}$ so we prove it by (K), or $\psi = \phi$ so we use $\vdash \phi \to \phi$ as derived above.
Case $n > 1$: ψ is the result of an application of modus ponens. We have χ and $\chi \to \psi$, both of which were derived from Γ, ϕ in fewer steps. Induction hypothesis gives us $\Gamma \vdash \phi \to \chi$ and $\Gamma \vdash \phi \to \chi \to \psi$. We use (S) in the form $(\phi \to \chi \to \psi) \to (\phi \to \chi) \to (\phi \to \psi)$ and apply modus ponens twice, resulting in $\phi \to \psi$ derived from Γ only.

Class 7

For this chapter, we replace the word *formula* in rules and proofs of a formal system by the word *judgement*.

0.6 Natural Deduction for Propositional Logic

Judgements in Natural Deduction have the form $\Gamma \vdash \phi$, where Γ is a set of formulas and ϕ is a formula. Semantically, for all interpretations v, if all formulas in Γ are true in v, then ϕ is true in v. Formal system for natural deduction defines a meta-symbol $\vdash_{\text{meta}}$ such that $\vdash_{\text{meta}} (\Gamma \vdash \phi)$.

0.6.1 Judgements in Natural Deduction

We use the notation Γ, ϕ to show $\Gamma \cup \{\phi\}$.

1. Axioms

$$\frac{}{\Gamma, \phi \vdash \phi}\ \text{AX} \qquad \frac{}{\Gamma \vdash \top}\ \top\text{-INTRO}$$

2. False-elimination: if $\bot$ can be derived, then anything can be derived.

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}\ \bot\text{-ELIM}$$

3. Conjunction elimination and introduction

$$\frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \wedge\text{-ELIM} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \wedge\text{-ELIM} \qquad \frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge\text{-INTRO}$$

4. Disjunction elimination and introduction

$$\frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma, \phi \vdash \chi \qquad \Gamma, \psi \vdash \chi}{\Gamma \vdash \chi}\ \vee\text{-ELIM} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \wedge \psi}\ \vee\text{-INTRO} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \psi \wedge \phi}\ \wedge\text{-INTRO}$$

5. Negation elimination and introduction

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \neg\phi}{\Gamma \vdash \bot}\ \neg\text{-ELIM} \qquad \frac{\Gamma, \phi \vdash \bot}{\Gamma \vdash \neg\phi}\ \neg\text{-INTRO}$$

6. Implication elimination and introduction

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \phi \to \psi}{\Gamma \vdash \psi} \to\text{-ELIM} \qquad\qquad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \to\text{-INTRO}$$

Observe how $\to$-elim is similar to modus ponens.

Homework 4. *Prove implication transitivity using Natural Deduction.*

Example 5. *Show* $(\phi \to \psi) \to (\neg\psi \to \neg\phi)$.

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\phi \to \psi, \neg\psi, \phi \vdash \psi \qquad \phi \to \psi, \neg\psi, \phi \vdash \neg\psi}{\phi \to \psi, \neg\psi, \phi \vdash \bot}\neg\text{-ELIM}}{\phi \to \psi, \neg\psi \vdash \neg\phi}\neg\text{-INTRO}}{(\phi \to \psi) \vdash (\neg\psi \to \neg\phi)}\to\text{-INTRO}}{\vdash (\phi \to \psi) \to (\neg\psi \to \neg\phi)}\to\text{-INTRO}$$

Now we have two goals to prove.

$$\dfrac{\dfrac{}{\phi \to \psi, \neg\psi, \phi \vdash \phi}\text{AX} \qquad \dfrac{}{\phi \to \psi, \neg\psi, \phi \vdash \phi \to \psi}\text{AX}}{\phi \to \psi, \neg\psi, \phi \vdash \psi}\to\text{-ELIM} \qquad \dfrac{}{\phi \to \psi, \neg\psi, \phi \vdash \neg\psi}\text{AX}$$

□

Observe how this method of writing proofs in Natural Deduction requires us to rewrite the context for every step. We can use a slightly different notation to avoid this repetition.

We can draw boxes to introduce *contexts* in a proof. Every formula written inside a box is assumed to hold only within that box. The following "proofs" are examples of using this notation.

$$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \to \psi} \qquad \dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \bot\end{array}}}{\neg\phi} \qquad \dfrac{\phi \vee \psi \qquad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}} \qquad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}}}{\chi}$$

Using this notation, we can rewrite the proof for example 5:

$$\boxed{\begin{array}{c}1.\ \phi \to \psi\\ \boxed{\begin{array}{c}2.\ \neg\psi\\ \boxed{\begin{array}{c}3.\ \phi\\ 4.\ \psi\ (\to\text{-elim, 1, 3})\\ 5.\ \bot\ (\neg\text{-elim, 2, 4})\end{array}}\\ \hline \neg\phi\end{array}}\\ \hline \neg\psi \to \neg\phi\end{array}}$$

$$\hline (\phi \to \psi) \to (\neg\psi \to \neg\phi)$$

The system defined above is in fact the NJ (Intuitionistic Natural Deduction) system. The following rule, namely the law of excluded middle, cannot be derived in NJ:

$$\frac{}{\Gamma \vdash \phi \vee \neg\phi}\ \text{EX-MIDDLE}$$

The NK system (Classical Natural Deduction) is NJ with the addition of the law of excluded middle. The NK system is sound and complete for propositional logic.

Assumption of the law of excluded middle is in fact an important distinction between Intuitionistic and classical logic. In the following is an example which uses excluded middle in its proof.

Example 6. *Show that there exist $a, b \notin \mathbb{Q}$ such that $a^b \in \mathbb{Q}$.*

Proof. Let $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. We know that $\sqrt{2} \notin \mathbb{Q}$. We do a "classical" case-splitting on $a \in \mathbb{Q}$:

1. $a \notin \mathbb{Q}$. We have
$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^2 = 2 \in \mathbb{Q}$$
2. $a \in \mathbb{Q}$. We are already done with the proof; let $a_1 = b_1 = \sqrt{2}$. We know $a_1, b_1 \notin \mathbb{Q}$ and, by assumption, ${a_1}^{b_1} \in \mathbb{Q}$.

Observe how this classical-style proof utilizes the law of excluded middle in the case-splitting: $\sqrt{2}^{\sqrt{2}}$ is either in $\mathbb{Q}$ or not in $\mathbb{Q}$; there is no *middle*. □

0.7 Kripke Semantics

Classically, an interpretation $v : P \to \mathbb{B}$ is defined as a mapping from a set of propositions to boolean values $\top$ and $\bot$. For intuitionistic reasoning, we define a new semantics.

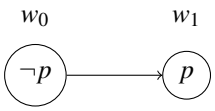
Definition 32. *A Kripke model m is defined as a tuple $(W, \leq, w_0, v : W \times P \to \mathbb{B})$, where W is a set of classical worlds, $\leq$ is a pre-order relation on W, w_0 is the initial world, and v is a function from pairs of world and proposition to boolean values such that for any $w, w' \in W$ and any $p \in P$, if $w \leq w'$, then $v(w,p) \leq v(w',p)$.*

Informally, a Kripke model is an interpretation model for intuitionistic proof systems. The following facts hold for any Kripke model m:

1. $m \models \phi$ iff $m \vdash_{w_0} \phi$.
2. $m \not\models_w \bot$ for any world w.
3. $m \models_w p$ iff $v(w,p) = \top$.
4. $m \models_w \phi \to \psi$ iff for any w', if $w \leq w'$ and $m \models_{w'} \phi$, then $m \models_{w'} \psi$.

In NJ, whenever you show $\phi \vee \psi$, you need to show either ϕ, or ψ. As previously stated, the law of excluded middle cannot be derived in NJ. To show this, we need to show that there exists a Kripke model m such that excluded middle is false in a world w of m.

Let us define a Kripke model with only one proposition p and only two worlds w_0 and w_1, where $w_0 \leq w_1$, and p is false in w_0 and true in w_1.



Let us examine what formulas are true (or false) in each world. By definition, p is false in w_0. Let us examine the value of $\neg p$ in w_0. We can safely substitute $\neg p$ with $p \to \bot$. By definition, $p \to \bot$ holds in w_0 iff for any world w, if $w_0 \leq w$ and p is true in w, then $\bot$ is true in w. We know p is true in w_1. We also know that $\bot$ is not true in w_1, as it is not true in any world. So, by definition, $p \to \bot$ is false in w_0. So, both p and $\neg p$ are false in w_0, from which we obtain that $p \vee \neg p$ is also false in w_0.

0.8 Sequent (Gentzen) Calculus and LK

Judgements in the LK proof system have the form $\Gamma \vdash \Delta$, where both Γ and Δ are sets of formulas. Judgement $\Gamma \vdash \Delta$ should be read as "the *conjunction* of the formulas in Γ implies the *disjunction* of the formulas in Δ". Semantically, for a classical interpretation v, $v \models (\Gamma \vdash \Delta)$ if and only if, if all formulas in Γ are true under v, then some fomula in Δ is true under v.

0.8.1 Judgements in LK

Observe how every non-axiom judgement increases the number of logical connectives in the set of formulas.

1. Axioms
$$\frac{}{\Gamma, \phi \vdash \phi, \Delta}\ \text{AX} \qquad \frac{}{\Gamma, \bot \vdash \Delta}\ \bot\text{-ELIM} \qquad \frac{}{\Gamma \vdash \top, \Delta}\ \top\text{-INTRO}$$

2. Conjunction
$$\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta} \qquad \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}$$

3. Disjunction
$$\frac{\Gamma, \phi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \vee \psi \vdash \Delta} \qquad \frac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \vee \psi, \Delta}$$

4. Negation
$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \neg\phi, \Delta} \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg\phi \vdash \Delta}$$

5. Implication
$$\frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \to \psi, \Delta} \qquad \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \to \psi \vdash \Delta}$$

The LK proof system is sound and complete for propositional logic. This actually means that excluded middle can be derived in LK.

Homework 5. *Prove the following in LK:*

1. *The law of excluded middle:* $\vdash p \vee \neg p$,
2. *Implication transitivity.*

Class 8

We now talk about three proof systems:

- Hilbert
$$\vdash \phi$$
- Natural deduction
$$\Gamma \vdash \phi$$
- Grentzen
$$\Gamma \vdash \Delta$$

We now talk abou three decision procedures (either for validity or for satisfiability):

- Branching (if we don't derive $\bot$ then it is satisfiable)
$$\frac{\Gamma[\bot] \qquad \Gamma[\top]}{\Gamma[p]}$$
- Resolution
$$\frac{\Gamma, C_1, C_2, C_1[\bot] \vee C_2[\bot]}{\Gamma, C_1[p], C_2[p]}$$
- Unit resolution, combined with branching of lower priority (DPLL)
$$\frac{\Gamma, C[\bot]}{\Gamma, \ell, C[\neg\ell]}$$

  Example:
$$p \vee q \vee r,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee q \vee r,\ \neg q \vee r,\ q \vee \neg r$$

  First we branch on r.

  - Case $r = \bot$:
$$p \vee q,\ \neg p \vee q,\ \neg q$$
    We didn't continue this branch.
  - Case $r = \top$:
$$\neg p \vee \neg q,\ q$$
    Then we propagate q. We end up with $\neg p$.
    It is satisfied by $p = \bot$, $q = \top$, $r = \top$.

Horn clause is a clause with at most one positive literal. We can view them as implications where LHS is a conjunction of positive propositions:

$$\begin{aligned} \neg p \vee \neg q &\iff p \wedge q \to \bot \\ \neg p \vee \neg q \vee r &\iff p \wedge q \to r \\ r &\iff \top \to r \end{aligned}$$

0.8.2 Metatheorems

Compactness

Countable set Γ of formulas is satisfiable iff every finite subset of Γ is satisfiable.

Craig's interpolation

We have $\vdash \phi \to \psi$ iff there exists a third formula χ (the "interpolant") which only uses nonlogical symbols (in propositional logic, it is propositions only) that occur in both ϕ and ψ such that $\vdash \phi \to \chi$ and $\vdash \chi \to \psi$.

Cut elimination

Cut rule in NK / NJ:

$$\frac{\Gamma \vdash \phi \qquad\qquad \Gamma, \phi \vdash \psi}{\Gamma \vdash \psi}$$

Cut rule in LK:

$$\frac{\Gamma \vdash \phi, \Delta \qquad\qquad \Gamma, \phi \vdash \Delta}{\Gamma \vdash \Delta}$$

If a judgement can be proved with the cut rule, it can also be proved without the cut rule. In fact, it is "iff"; the other direction is trivial. Of course, the proof may become longer (introducing a lemma ϕ often helps in practice). It is a purely syntactic metatheorem. We cannot use deduction to prove it.

0.9 First-order logic

First-order logic, also called "predicate logic", is propositional logic with quantifiers $\forall$ (for all) and $\exists$ (exists).

0.9.1 Syntax

Nonlogical symbols ("signature")

- finite set of variables $X = \{x, y, z, \dots\}$
- finite set of function symbols $F = \{f, g, h, \dots\}$
- finite set of predicate symbols $P = \{p, q, r, \dots\}$

Each function symbol has a fixed "arity" (number of arguments), which can be zero (arity 0 gives a constant). Each predicate symbol has also a fixed "arity" (number of arguments), which can be zero (arity 0 gives a proposition).

Logical symbols

- connectives ($\bot, \top, \neg, \wedge, \vee, \rightarrow$)
- quantifiers ($\forall, \exists$)
- finite set of predicate symbols $P = \{p, q, r, \dots\}$

We don't add parentheses to the symbols; we will talk about syntax trees; only if we want to write them down as strings, we add parentheses (as few as possible).

Grammar

- Terms: $\phi \;:=\; f_0 \mid f_n(t_1, \dots, t_n)$
- Formulas: $\phi \;:=\; p_0 \mid p_n(t_1, \dots, t_n) \mid \bot \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \forall x.\phi \mid \exists x.\phi$

When we write $\forall x.\phi$, every occurence of x is bound in ϕ. A variable that is not bound is called free. A change of bound variables does not change the abstract syntax tree (same in abstract syntax).

Safe substitution

$$\forall x.\ \exists y.\ y \geq x+1$$

If we want to substitute y^2 for x, we must first rename y to z and then substitute.

$$\forall x.\ \exists z.\ z \geq x+1$$

$$\exists z.\ z \geq y^2+1$$

First Order Logic (FOL)

Definition 33. *Interpretation I in first-order structure:*

1. *Domain* D_I
2. *for each n-ary function symbol* $f \in F$, $[f]_I : D_I^n \to D_I$
3. *for each n-ary predicate symbol* $p \in P$, $[p]_I : D_I^n \to B = \{true, false\}$
4. *"context" (environment) for each (free) variable* $x \in X$, $[x]_I \in D_I$

Formula ϕ is closed if it has no free variables.
Given an interpretation I, a term t and a formula ϕ have the following meaning:

$$[t]_I = \begin{cases} [x]_I & t = x \\ [f]_I([t_1]_I, [t_2]_I, \ldots, [t_n]_I) & t = f(t_1, \ldots, t_n) \end{cases}$$

$$[\phi]_I = \begin{cases} [x]_I & t = x \\ [p]_I([t_1]_I, [t_2]_I, \ldots, [t_n]_I) & \phi = p(t_1, \ldots, t_n) \end{cases}$$

$$[\phi_1 \Rightarrow \phi_2]_I = True \text{ iff} [\phi_1]_I = false \text{ or } [\phi_2]_I = true$$

$$[\forall x.\phi]_I = True \text{ iff for all } d \in D_I, [\phi]_{I[x \to d]}$$
$$[\exists x.\phi]_I = True \text{ iff for some } d \in D_I, [\phi]_{I[x \to d]}$$

$$\phi \text{ valid } (\vDash \phi) \text{ iff } [\phi]_I = True \text{ for all interpretations } I$$

$$\phi \text{ satisfiable iff } [\phi]_I = True \text{ for some interpretations } I$$

0.9.2 Small Detour

PCP (Post Correspondence Problem): Given a finite set S of dominoes $\frac{s}{t}$ where $s, t \in \{0,1\}^*$. Is there a finite sequence $\frac{s_1}{t_1}, \ldots \frac{s_n}{t_n}$ of (possibly repeating) dominoes from S such that the

$$s_1 \cdot s_2 \cdot \cdots \cdot s_n = t_1 \cdot t_2 \cdot \cdots t_n$$

Theorem 10. *The PCP problem is undecidable.*

0.9.3 Back to Work

Metatheorem 1. *1. (Compactness) Set Γ of formulas is satisfiable iff every finite subset of Γ is satisfiable.*

2. (Lowenheim-Skolem). If a set Γ of formulas is satisfiable then Γ is satisfiable by an interpretation with countable domain.

3. Both the validity and satisfiability problems for FOL are undecidable.

Proof. (Proof sketch of undecidability) Let $F = \{e, f_0, f_1\}$ where e is 0-ary and f_0, f_1 are unary. Basically a string 011 of 0,1s is represented as $f_1(f_1(f_0(e)))$.

Let $P = \{p\}$ where p is a binary predicate. Basically a domino $\frac{s}{t}$ is represented by $p(s,t)$.

Given an instance R of PCP, the formula $\phi_R = (\phi_1 \wedge \phi_2) \Rightarrow \phi_3$ is valid iff the answer to R is yes:

$$\phi_1 = \bigwedge_{1 \le i \le k} P(s_i(e), t_i(e))$$
$$\phi_2 = \forall v, w.(p(v,w) \Rightarrow \bigwedge_{1 \le i \le k} p(s_i(v), t_i(w)))$$
$$\phi_3 = \exists z.p(z,z)$$

□

0.9.4 Three Sound and Complete Proof Systems

Hilbert

$$1.(\forall x.\phi) \Rightarrow \phi[x := t]$$
$$2.\forall x.(\phi \Rightarrow \psi) \Rightarrow (\forall x.\phi) \Rightarrow (\forall x.\psi)$$
$$3.\phi \Rightarrow \forall x.\phi \quad \textit{provided that x is not free in } \phi$$

($\phi[x := t]$ means safely replacing each occurrence of x by t)

0.9.5 Gentzen

$$\frac{\Gamma, \phi[x := t] \vdash \Delta}{\Gamma, \forall x.\phi \vdash \Gamma} \forall - elim$$
$$\frac{\Gamma \vdash \Delta, \phi[x := y]}{\Gamma \vdash \Delta, \forall x.\phi} \forall - intro \; y \textit{ is a new (fresh) variable}$$
$$\frac{\Gamma, \phi[x := y]}{\Gamma, \exists x.\phi \vdash \Delta} \exists - elim$$
$$\frac{\Gamma \vdash \Delta, \phi[x := t]}{\Gamma \vdash \Delta, \exists x.\phi} \exists - intro$$

0.9.6 Natural Deduction

$$\frac{\forall x\phi}{\phi[x:=t]} \qquad \frac{\phi[x:=t]}{\exists x.\phi}$$

$$\frac{\begin{bmatrix} y \text{ new} \\ \dots \\ \phi[x:=y] \end{bmatrix}}{\forall x.\phi} \qquad \frac{\exists x.\phi, \begin{bmatrix} y:\phi[x:=y] \\ \dots \\ \psi \end{bmatrix}}{\psi}$$

Homework 6. *Prove de Morgan for quantifiers*

$$\forall x.\phi \Leftrightarrow \nexists x.\ \phi$$

in all three systems.

Homework 7. *Use PCP to show satisfiability of FOL is undecidable.*

0.9.7 First Order Resolution: Classical Automated Theorem Proving

We want to prove $\models \phi$ for a closed ϕ.

1. Negate. Show $\neg\phi$ is unsat.
2. Bring $\neg\phi$ into "prenex" form (all quantifiers are in front). (Ex. $\exists x.\phi(x) \wedge \gamma \iff \exists x(\phi(x) \wedge \gamma)$).
3. Bring γ into CNF.
4. "Skolemization" gets rid of $\exists$.
   - Consider that while for $\exists x.\phi(x)$ is sat iff $\phi(\hat{x})$ is sat, we have that $\forall y \exists x \phi(x)$ is sat iff $\phi(f(y))$ is sat where $f(y)$ is a new function symbol. Thus,for each $\exists x$ within the scope of a universally quantified variables $y_1, \cdots, y_n$, drop $\exists x$ and replace each bound occurrence of x by $f(y_1, \cdots, y_n)$ where f is a new n-ary function symbol. E.g.

$$\forall x, \exists y \forall z_1, z_2 \exists y.\phi(x,y,z_1,z_2,u) \rightarrow \forall x \forall z_1, z_2.\phi(x, f(x), z_1, z_2, g(x,z_1,z_2)). \tag{9}$$

5. Drop $\forall$.
6. *Theorem.* For an interpretation I, clauses C_1 and C_2, and literals l and l'. If $I \models C_1[l]$ and $I \models C_2[\neg l']$ and l, l' are "unifiable" (i.e, there is a substitution, which is a function from variables to terms, that when applied to l and l', makes them equal), then $I \models X_1\theta[\perp] \vee C_2\theta[\perp]$ where θ is the most general unifier of l and l'.

   For example for a variable x and a constant S, we have that $m(x)$ and $\neg m(S)$ are unifiable by replacing x by s. The purpose is to make literals the same.

0.9.8 First order theories

Theories defined by 1) signature (set of functions + predicate symbols) and 2) either by a r.e. set of closed formulas called axioms A (set of all A-valid closed formulas) or by a specific "intended" interpretation I (true formulas in I).

Formally, a theory T is a r.e. set of closed formulas (there are no free variables) that are closed under $\models$, i.e

Def. $T \models \phi$ iff $\forall I((\forall \psi \in T . I \models \psi) \Rightarrow I \models \phi)$.

We say that

- ϕ is $T-$valid iff $T \models \phi$
- ϕ is $T-$satisfiable iff $\exists I . I$ T-model ϕ.
- I is a $T-$model iff $\forall \psi \in T . I \models \psi$.
- ϕ, ψ are $T-$equivalent iff ϕ, ψ have truth value in all $T-$models.

Therefore, T is either (1) the set of all $A-$valid closed formulas or (2) T is the set of all formulas true in I.

T is consistent iff $\exists I . \forall \psi \in T, I \models \psi$ (always the case for (2)).

T is complete iff for all closed formulas ψ of the signature, either $T \models \psi$ or $T \models \neg \psi$ (always the case for (2)).

T is decidable iff the problem "given ψ, is $T \models \psi$" is decidable.

Class 16

Theorem 11 (Soundness)**.** *If* $\{\phi\} \vdash \{\psi\}$ *then* $\{\phi\} \models \{\psi\}$ *such that for*

- *partial correctness programs we have that*

$$\forall \sigma \sigma' \forall I,\ \sigma \models^I \phi \wedge \langle c, \sigma \rangle \downarrow \sigma' \Rightarrow \sigma' \models^I \psi,$$

- *and for total correctness programs we have that*

$$\forall \sigma \forall I,\ \sigma \models^I \phi \Rightarrow \exists \sigma', \langle c, \sigma \rangle \downarrow \sigma' \wedge \sigma' \models^I \psi.$$

Theorem 12 (Relative Completeness)**.** *We have an oracle for deciding* $\models \chi$ *(the validity of assertions), if* $\models \{\phi\} c \{\psi\}$ *then* $\vdash \{\phi\} c \{\psi\}$.

However, the following issues arise:

- $\models \{true\} c \{false\}$ `iff` c does not terminate.
- $\models \{true\} skip \{\psi\}$ `iff` $\models \psi$, but according to Gödel theorem there is no complete proof system for $(\mathbb{N}, +, \times)$.

0.10 Verification Conditions

In general, we can derive verification conditions using the following rules:

- $vc(\{\phi\} skip \{\psi\}) = \{\phi \Rightarrow \psi\}$.
- $vc(\{\phi\} x = a \{\psi\}) = \{\phi \Rightarrow \psi_{[x \mapsto a]}\}$.
- $vc(\{\phi\} c_1 \{\chi\} c_2 \{\psi\}) = vc(\{\phi\} c_1 \{\chi\}) \cup vc(\{\chi\} c_2 \{\psi\})$.
- $vc(\{\phi\}$ `if` b `then` c_1 `else` $c_2 \{\psi\}) = vc(\{\phi \wedge b\} c_1 \{\psi\}) \cup vc(\{\phi \wedge \neg b\} c_2 \{\psi\})$.
- $vc(\{\phi\}$ `while` $\{\chi\}\ b$ `do` $c \{\psi\}) = \{\phi \Rightarrow \chi, \chi \wedge \neg b \Rightarrow \psi\} \cup vc(\{\chi \wedge b\} c \{\chi\})$.

0.10.1 Annotated Program

From the annotations shown in Algorithm 1 we can derive the verification conditions which are the assertions that need to be valid for the Hoare proof:

- $true \Rightarrow 0 = 0$.

Algorithm 1 A program that computes $m \times n$

$\{true\}$
$x \coloneqq 0$
$\{x = 0\}$
$y \coloneqq 0$
$\{x = 0 \wedge y = 0\}$
while $y \leq n$ **do** ▷ loop invariant: $\{x = m \times y \wedge y \leq n\}$
  $\{x = m \times y \wedge y < n\}$
  $x \coloneqq x + m$
  $\{x = m \times (y+1) \wedge y < n\}$
  $y \coloneqq y + 1$
end while
$\{x = m \times n\}$

- $x = 0 \Rightarrow x = 0 \wedge 0 = 0$.
- $x = 0 \wedge y = 0 \Rightarrow x = m \times y \wedge y \leq n$.
- $x = m \times y \wedge y \leq n \wedge y \geq n \Rightarrow x = m \times n$.
- $x = m \times y \wedge y \leq n \wedge y < n \Rightarrow x = m \times y \wedge y < n$.
- $x = m \times y \wedge y < n \Rightarrow x + m = m \times (y+1) \wedge y < n$.
- $x = m \times (y+1) \wedge y < n \Rightarrow x = m \times (y+1) \wedge (y+1) \leq n$.

0.10.2 Flow Chart (Control Flow Graph - "Floyd Style")

The annotations for the same example can be seen in Figure 0.10.2.

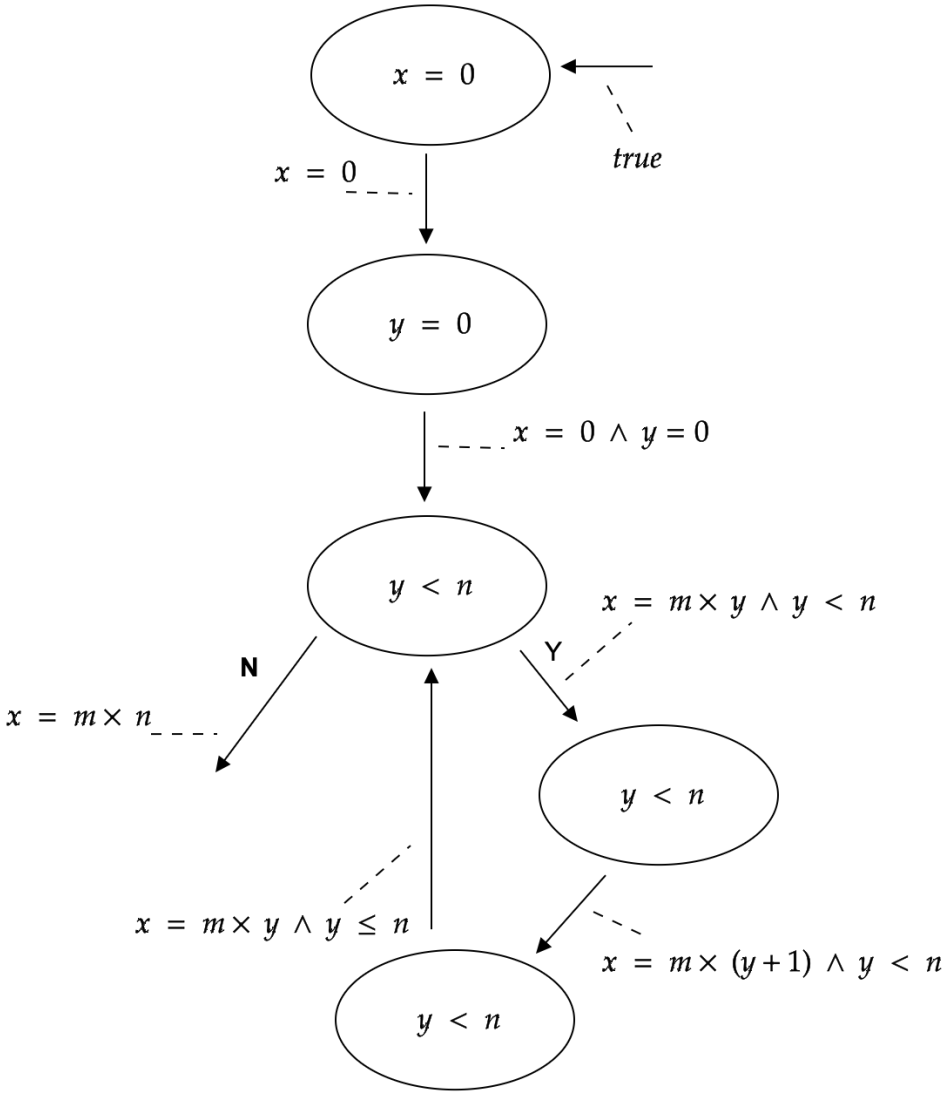


Figure 1: Control flow graph of the program that computes $x = m \times n$.

0.11 Programs as predicate transforming (Dijkstra): Weakest (liberal) preconditions

Definition 34 (Weakest Liberal Preconditions)**.** *$wlp(c,\psi)$ is the weakest ϕ s.t $\{\phi\}c\{\psi\}$ (for all ϕ with $\{\phi\}c\{\psi\}, \phi \Rightarrow wlp(c,\psi)$).*

Definition 35 (Weakest Preconditions)**.** *$wp(c,\psi)$ is the weakest ϕ s.t $\{\phi\}c\{\psi\} \wedge \forall\sigma$, if $\sigma \models \phi$ then c terminates from σ.*

An assertion language is expressive if $wlp(c,\psi) = \{\sigma \in \Sigma \mid \langle c,\sigma\rangle = \bot \vee \forall I.[\![c]\!]\sigma \models^I \psi\}$.
For IMP, arithmetic $(\mathbb{N},+,\times)$ is expressive (proof by Gödelization of program executions).

Example 7.

- $wlp(skip,\psi) = \psi$
- $wlp(x = a,\psi) = \psi_{[x\mapsto a]}$
- $wlp(c_1,c_2,\psi) = wlp(c_1,wlp(c_2,\psi))$
- $wlp(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\psi) = (b \Rightarrow wlp(c_1,\psi)) \wedge (\neg b \Rightarrow wlp(c_2,\psi)) = (b \wedge wlp(c_1,\psi)) \vee (\neg b \wedge wlp(c_2,\psi))$.
- $wlp(\texttt{while } b \texttt{ do } c,\psi) = \texttt{weakest } \chi$ *s.t 1)* $\chi \wedge b \Rightarrow wlp(c,\chi)$ *and 2)* $\chi \wedge \neg b \Rightarrow \psi$.
- *For a fresh variable n,* $wp(\texttt{while } b \texttt{ do } c,\psi) = \texttt{weakest } \chi$ *s.t for a fresh variable n 1)* $\chi \wedge b \wedge e = n \Rightarrow wlp(c,\chi \wedge e > n)$*, 2)* $\chi \wedge \neg b \Rightarrow \psi$*, and 3)* $\chi \Rightarrow e \in$ *well founded set.*

Class 17

$$[\![\text{skip}]\!] = \lambda\sigma.\sigma$$

$$[\![\text{assert } b]\!] = \lambda\sigma.\text{ if } [\![b]\!]\sigma \text{ then } \sigma \text{ else fail}$$

$$[\![c]\!] : \Sigma \to \Sigma_\perp$$

$$[\![c]\!] : \Sigma \to \Sigma_\perp^{\text{fail}} = \Sigma \cup \{\text{fail}, \perp\}$$

Operationally:

$$\frac{\langle b, \sigma\rangle \downarrow \top}{\langle \text{assert } b, \sigma\rangle \downarrow \sigma}$$

In nondeterministic settings, the symbol $\downarrow$ means "can reach".

$$\frac{\langle b, \sigma\rangle \downarrow \perp}{\langle \text{assert } b, \sigma\rangle \downarrow \text{fail}}$$

$$\text{wlp}(\text{skip}, \psi) = \psi$$

$$\text{wlp}(\text{assert } b, \psi) = b \wedge \psi$$

$$\{b \wedge \psi\} \text{ assert } b \{\psi\}$$

$$[\![c]\!] \subseteq \Sigma \times \Sigma_\perp^{\text{fail}}$$

$$[\![\text{assume } b]\!] = \{(\sigma, \sigma') \mid \sigma \vDash b \wedge \sigma' = \sigma\}$$

We define:

$$\text{wlp}(c, \psi) = \psi = \{\sigma \mid \forall\sigma'.(\sigma, \sigma') \in [\![c]\!] \implies \sigma' \vDash \psi\}$$

$$\text{wlp}(\text{assume } b, \psi) = b \implies \psi$$

$$\{b \implies \psi\} \text{ assume } b \{\psi\}$$

Assert false:

$$[\![\text{abort}]\!] = \lambda\sigma.\text{fail}$$

$$[\![\text{havoc } b]\!] = \{(\sigma, \sigma') \mid \sigma' \vDash b\}$$

0.12 Disjkstra's guarded commands

$$\text{gc} := b \to c \mid \text{gc} \,[]\, \text{gc}$$

We say b is a "guard" and we say $[]$ means "alternatively". For example:

$$x \leq 5 \to x := x+1 \,[]\, x \geq 5 \to x := x-1$$

$$c := \text{skip} \mid \text{abort} \mid x := a \mid c;c \mid \text{if gc fi} \mid \text{do gc od}$$

$$\frac{\langle b, \sigma\rangle \downarrow \top \qquad \langle c, \sigma\rangle \downarrow \sigma'}{\langle b \to c, \sigma\rangle \downarrow \sigma'}$$

$$\frac{\langle b, \sigma\rangle \downarrow \bot}{\langle b \to c, \sigma\rangle \downarrow \text{fail}}$$

$$\frac{\langle \text{gc}_1, \sigma\rangle \downarrow \sigma'}{\langle \text{gc}_1 \,[]\, \text{gc}_2, \sigma\rangle \downarrow \sigma'}$$

$$\frac{\langle \text{gc}_2, \sigma\rangle \downarrow \sigma'}{\langle \text{gc}_1 \,[]\, \text{gc}_2, \sigma\rangle \downarrow \sigma'}$$

$$\frac{\langle \text{gc}_1, \sigma\rangle \downarrow \text{fail} \qquad \langle \text{gc}_2, \sigma\rangle \downarrow \text{fail}}{\langle \text{gc}_1 \,[]\, \text{gc}_2, \sigma\rangle \downarrow \sigma'\text{fail}}$$

$$\frac{\langle \text{gc}, \sigma\rangle \downarrow \sigma'}{\langle \text{if gc fi}, \sigma\rangle \downarrow \sigma'}$$

$$\frac{\langle \text{gc}, \sigma\rangle \downarrow \text{fail}}{\langle \text{if gc fi}, \sigma\rangle \downarrow \text{fail}}$$

$$\frac{\langle \text{gc}, \sigma\rangle \downarrow \text{fail}}{\langle \text{do gc od}, \sigma\rangle \downarrow \sigma}$$

$$\frac{\langle \text{gc}, \sigma\rangle \downarrow \sigma' \qquad \langle \text{do gc od}, \sigma'\rangle \downarrow \sigma''}{\langle \text{do gc od}, \sigma\rangle \downarrow \sigma''}$$

Euclid(gcd):

$$\text{do } x > y \to x := x-y \,[]\, y > x \to y := y-x \text{ od}$$

0.13 Parallelism

For example, $x := 0 \;||\; x := 1$ is either $(x := 0;\ x := 1)$ or $(x := 1;\ x := 0)$.

Parallelism as nondeterministic interleaving of atomic actions (programs):

1. nondeterministic
2. noncompositional
3. nonterminating "reactive programs"

Concurrency Theory / Process Algebra (we want to add || to languages). There were dozens of process algebras in the 1990s.

0.13.1 CCS (Calculus of Communicating Systems) by Milner

- processes: P, Q, ...
- nil: 0 (like skip)
- atomic actions: $a, b, \ldots, \in A$
- prefix: $a.P$ (like sequencing)
- alternative: $P + Q$
- process def: $C \triangleq P$
- parallelism: $P||Q$
- restriction: $(\nu a)P$

Iterate = a.Iterate + b 0

0.13.2 SOS (Structured Operational Semantics)

(single step)

$$\frac{}{a.P \xrightarrow{a} P}$$

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$$

$$\frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} P'}{C \xrightarrow{a} P'}\ C \triangleq P$$

$$\frac{P \xrightarrow{a} P'}{P||Q \xrightarrow{a} P'||Q}$$

$$\frac{Q \xrightarrow{a} Q'}{P||Q \xrightarrow{a} P||Q'}$$

Synchronized communication:

$$P \xrightarrow{a} P' \qquad Q \xrightarrow{\bar{a}} Q'$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\bar{a}} Q'}{P||Q \xrightarrow{\tau} P'||Q'}$$

$$\frac{P \xrightarrow{b} P'}{(\nu a)P \xrightarrow{b} (\nu a)P'} \; a \neq b, \bar{a} \neq b$$

Class 22

Calculus of Communicating Sysfems (CCS) can be given two semantics.

- *Sequential.* Labeled transition systems and logics (e.g., Hennessy-Milner logic (HML) and temporal logic)
- *Truly concurrent.* Parallelism is different from nondeterminism. Uses Petri nets.

Definition 36 (Petri nets)**.** *A* petri net *is a tuple $N = (P,T)$ where P is a set of places and T a set of transitions. The state of a petri net is given by a marking $M\colon P \to \mathbb{N}$ and a transition is a function that takes a marking and an action and returns another marking, i.e., $t\colon M \to M$. Equivalently, considering a set of names for each transition (actions) A, we have that $T : M \times A \to M$.*

Remark 1 (Petri nets as lebeled transition systems)**.** *A Petri net $N = (P,T)$ can be interpreted as the following labeled transition system (LTS) $LTS_N = (M, \to)$. The transition $\to$ is given by $m \to m'$ if and only if there exists $t \in T$ such that*

- *the transformation t decomposes in precodition ${}^*t \in M$, action $a \in A$ and postcondition $t^* \in M$, i.e., $t = ({}^*t, a, t^*)$.*
- *the precodition ${}^*t$ is contained in m, i.e., ${}^*t \subseteq m$.*
- *the transformation t converts m into m', i.e., $m' = m -{}^* t + t^*$.*

Example 8 (Petri net)**.** *Consider the Petri net $N = (P,T)$ where $P = \{p_1, p_2, p_3, p_4, p_5\}$, $T = \{t\}$, and $t\colon \{(p_1,2),(p_2,1)\} \mapsto \{(p_4,1),(p_5,1)\}$. In other words, t can fire a transition if there are 2 markings in p_1 and 1 marking in p_2, and adds 1 marking in p_4 and 1 marking in p_5.*

Definition 37 (Sequenctial nets)**.** *A Petri net is* sequential *if all transformations t have a precondition ${}^*t$ such that $|{}^*t| \sqsubseteq 1$. Sequential nets are equivalent to regular laguages for finite strings.*

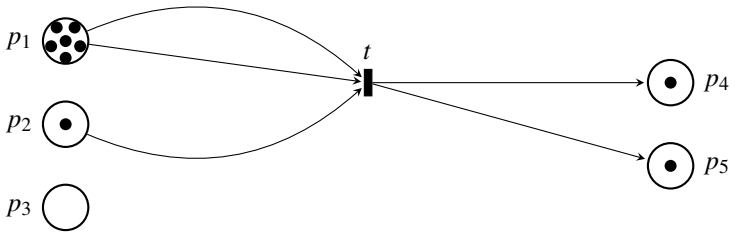


Figure 2: Petri net example

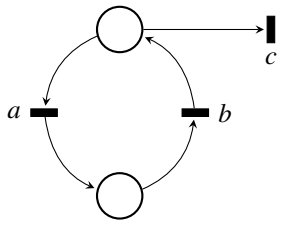


Figure 3: Sequential net example

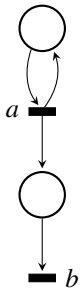


Figure 4: Basic parallel process example

Definition 38 (Basic parallel processes)**.** *A Petri net is a* basic parallel processes *if all transformations t have a precondition ${}^*t$ such that $|{}^*t| = 1$. Basic parallel processes are equivalent to context-free languages for finite strings.*

Definition 39 (Chomsky hierarchy)**.** *The* Chomsky hierarchy *is a hierarchy of sequential computation given by*

$$FA \subseteq PDA \subseteq LBA \subseteq TM\,,$$

where the decidabilities are as follows.

- *FA. Fully decidable.*
- *PDA. Universality is undecidable, i.e., $L(A) \stackrel{?}{=} \Sigma^*$.*
- *LBA. Emptyness is undecidable, i.e., $L(A) \stackrel{?}{=} \emptyset$.*
- *TM. Membership is undecidable, i.e., $w \stackrel{?}{\in} L(A)$.*

Definition 40 (Concurrency hierarchy)**.** *The* concurrency hierarchy *is a hierarchy of concurrent computation given by*

$$FA \subseteq BPP \subseteq CCS \subseteq TM\,,$$

where, in CCS, reachability is decidable but non-elementary.

Homework 8 (Ortogonal hierarchies)**.** *Solve the following tasks.*

1. *Show that $X = a(X|b) + c(X|d) + e$ is not context-free. Hint: use the pumping lemma.*
2. *Present a context-free language that is not in BPP, and try to prove it.*

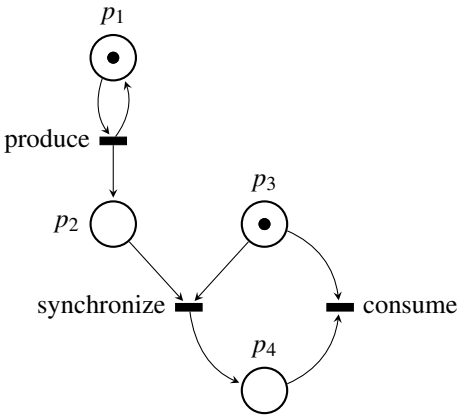


Figure 5: Karp-Miller Petri net with initial configuration example

3. Give a Petri net that generates $\{a^n b^n | n > 0\}$.

Definition 41 (Coverage problem for Petri nets)**.** *Given a Petrin net N and two marking* m, m'*, the problem is to decide the existence of a sequence of transitions that take m to a marking that covers* m'*, i.e., if there exists* $m*$ *such that* $m \to m*$ *and* $m' \subseteq m*$.

Remark 2. *As opposed to the reachability question, the coverage problem asks for* $m' \subseteq m*$ *instead of* $m' = m*$.

The coverage problem is a finite problem, since every finite Petri net N has a finite coverage tree of size $\mathcal{O}(n^n)$ where $n = |N|$. The coverage tree can be obtained by the Karp-Miller algorithm, which generates a tree of depth at most $\mathcal{O}(n \log n)$.

Consider the Petri net with an initial marking is given in Figure 5. The corresponding coverage tree is given in Figure 6. Note that m' is covered by m if there is a node in the tree that covers it.

Definition 42 (Control-flow models)**.** *Graphs whose nodes represent control states and whose edges represent control flow. For example, labeled transition systems and Petri nets.*

Definition 43 (Data-flow models)**.** *Also called circuits, they are graphs whose nodes represent actors (gates) and whose edges represent data flow.*

Figure 7 represents a data-flow model.

Definition 44 (Combinatorial circuits)**.** *A data-flow model is a* combinatorial circuit *if there are no loops. For combinatorial circuits, every tuple of inputs produces a tuple of outputs.*

Definition 45 (Sequential circuits)**.** *A data-flow model is a* sequential circuit *if there are loops. For sequential circuits, every sequence of inputs produces a sequence of outputs.*

To deal with loops in sequential circuits, there are two approaches:

- synchronous (clocked); and
- asynchronous (queues as channels).

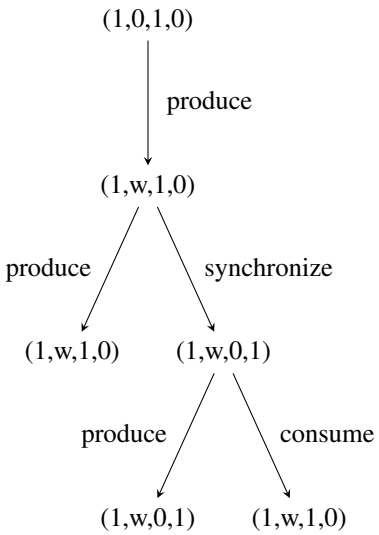


Figure 6: Karp-Miller coverage tree

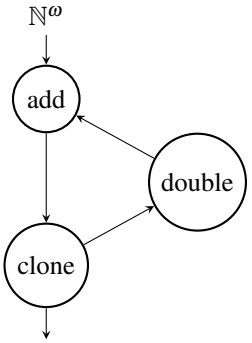


Figure 7: Data-flow model example