Formal Methods for Security and Privacy (SS 2020) Univ-Prof. Dr. Matteo Maffei

Final Exam

Date: 01/07/2020 TU Wien

- The exam takes 180 minutes. You can get at most 100 points.
- Your solutions can be handwritten and then scanned or photographed, or directly typed up.
  - Upload a single file (either a single pdf, or a zip archive containing multiple images (jpeg, png)) to the TUWEL assignment.
  - Make sure that the result is legible.
  - Your file may not be bigger than 256MB.
  - Your solution must be handed in **before 12:00 on the day of the exam**.
- **You must work on the exam alone.** You may use available resources (for example lecture slides), but **you must not communicate with anyone** except the lecturers. Everything you hand in must be written and created by you and you must be able to explain your solution. You will be required to confirm this when you upload your file. If plagiarism is detected, all of the parties involved (both commiting and aiding) will be held accountable.
- There will be an oral examination tomorrow (02/07/2020), where we will ask you to explain your solutions. Note however, that **your written solutions must be entirely self-contained**. The oral examination serves to verify that you created your solution on your own.
- In case of questions or uncertainties, you can call the phone number +43 (1) 58801-184864, where one of the lecturers will answer your questions.
  This is the only allowed way of communication during the exam.
- In case we should have an announcement that is relevant to all participants, we will post it in the TUWEL discussion board `https://tuwel.tuwien.ac.at/mod/forum/view.php?id=869944`. Please make sure that you receive these announcements.
- **Write legibly and be concise**. It is in your best interest that we understand your answers. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it.
- Please make sure your name and matriculation number are written on the document you hand in.
- Good luck!

Problem	1	2	3	4	**Total**
Points	30	20	20	30	100

$$
\begin{array}{rl}
\ell_C, \ell_I & ::= H \mid L \\
\ell & ::= \ell_C \ell_I \\
T & ::= \ell \mid \mathrm{C}^{\ell}[\tilde{T}] \mid \mathrm{SymK}^{\ell}[\tilde{T}]
\end{array}
$$

$$
\begin{array}{rl}
L & \sqsubseteq_C H \\
H & \sqsubseteq_I L \\
\ell_C^1 \ell_I^1 & \sqsubseteq \ell_C^2 \ell_I^2 \iff \ell_C^1 \sqsubseteq_C \ell_C^2 \wedge \ell_I^1 \sqsubseteq_I \ell_I^2
\end{array}
$$

$$
\begin{array}{rl}
\ell_1 & \leq \ell_2 \text{ whenever } \ell_1 \sqsubseteq \ell_2 \\
LL & \leq \mathrm{C}^{LL}[LL, \cdots, LL] \\
\mathrm{C}^{\ell}[\tilde{T}] & \leq \ell \\
LL & \leq \mathrm{SymK}^{LL}[LL, \cdots, LL] \\
\mathrm{SymK}^{\ell}[\tilde{T}] & \leq \ell
\end{array}
$$

$$
\begin{array}{rl}
\mathcal{L}(\ell) & = \ell \\
\mathcal{L}(\mathrm{C}^{\ell}[\tilde{T}]) & = \ell \\
\mathcal{L}(\mathrm{SymK}^{\ell}[\tilde{T}]) & = \ell \\
\mathcal{L}_{I,\Gamma}(\{|M|\}_k^s) & = \begin{cases} H & \text{if } \Gamma(K) = \mathrm{SymK}^{HH}[T] \\ & \quad \wedge \mathcal{L}_{I,\Gamma}(M) \sqsubseteq_I \mathcal{L}_I(T) \\ L & \text{otherwise} \end{cases} \\
\mathcal{L}_C(\ell_C \ell_I) & = \ell_C \\
\mathcal{L}_I(\ell_C \ell_I) & = \ell_I \\
\mathcal{L}_{\Gamma}(M) & = \begin{cases} \mathcal{L}(\Gamma(M)) & \text{if } M \in dom(\Gamma) \\ LL & \text{otherwise} \end{cases}
\end{array}
$$

$$
\textsc{Empty}\ \frac{}{\emptyset \vdash \diamond} \qquad \textsc{Env}\ \frac{\Gamma \vdash \diamond \qquad M \notin dom(\Gamma) \qquad T \in \{\mathrm{C}^{\ell}[\tilde{T}], \mathrm{SymK}^{\ell}[\tilde{T}]\} \text{ implies } \ell = HH}{\Gamma, M : T \vdash \diamond}
$$

$$
\textsc{Atom}\ \frac{\Gamma \vdash \diamond \qquad M : T \text{ in } \Gamma}{\Gamma \vdash M : T} \qquad \textsc{List}\ \frac{\Gamma \vdash M_1 : T_1 \qquad \ldots \qquad \Gamma \vdash M_n : T_n}{\Gamma \vdash [M_1, \ldots, M_n] : [T_1, \ldots, T_n]}
$$

$$
\textsc{Subsumption}\ \frac{\Gamma \vdash M : T' \qquad T' \leq T}{\Gamma \vdash M : T} \qquad \textsc{SymEnc}\ \frac{\Gamma \vdash K : \mathrm{SymK}^{\ell_C \ell_I}[\tilde{T}] \qquad \Gamma \vdash \tilde{M} : \tilde{T}}{\Gamma \vdash \{|\tilde{M}|\}_K^s : L\ell_I}
$$

$$
\textsc{Stop}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}} \qquad \textsc{Par}\ \frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \textsc{Repl}\ \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \textsc{Res}\ \frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a : T)\ P}
$$

$$
\textsc{Cond}\ \frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : T' \qquad \Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash \text{if } M = N \text{ then } P \text{ else } Q} \qquad \textsc{In}\ \frac{\Gamma, \tilde{x} : \tilde{T} \vdash P \qquad \Gamma \vdash N : \mathrm{C}^{\ell}[\tilde{T}]}{\Gamma \vdash N(\tilde{x}).P}
$$

$$
\textsc{Out}\ \frac{\Gamma \vdash \tilde{M} : \tilde{T} \qquad \Gamma \vdash P \qquad \Gamma \vdash N : \mathrm{C}^{\ell}[\tilde{T}]}{\Gamma \vdash \overline{N}\langle \tilde{M} \rangle.P} \qquad \textsc{SymDec}\ \frac{\Gamma \vdash M : T \qquad \Gamma \vdash K : \mathrm{SymK}^{\ell}[\tilde{T}] \qquad \Gamma, \tilde{x} : \tilde{T} \vdash P}{\Gamma \vdash \text{ case } M \text{ of } \{|\tilde{x}|\}_K^s \text{ in } P}
$$

Definition 1 (Opponent). *A process O is an opponent if any $(\nu a : T)$ ocurring in O are such that $T = LL$.*

Definition 2 (Secrecy). *P preserves secrecy if, for all opponents O, whenever $P \mid O \to^* (\nu c : T)\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle c \rangle.P'')$ we have $\mathcal{L}_C(T) \sqsubseteq_C \mathcal{L}_{C,\Gamma}(b)$, with $\Gamma = c : T, \tilde{a} : \tilde{T}$.*

Definition 3 (Integrity). *P preserves integrity if, for all opponents O, whenever*

$$
\begin{array}{l}
P \mid O \to^* (\nu b : C^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \overline{b}\langle c \rangle.P'') \text{ or} \\
P \mid O \to^* (\nu k : SymK^{HH}[T])\ (\nu \tilde{a} : \tilde{T})\ (P' \mid \text{case } \{|c|\}_k^s \text{ of } \{|x|\}_k^s \text{ in } P'')
\end{array}
$$

we have $\mathcal{L}_{I,\Gamma}(c) \sqsubseteq_C \mathcal{L}_I(T)$, with $\Gamma = b : C^{HH}[T], k : SymK^{HH}[T], \tilde{a} : \tilde{T}$.

Theorem 1 (Typing implies Secrecy and Integrity). *Let $\Gamma \vdash P$ with $img(\Gamma) = LL$. Then P preserves both secrecy and integrity.*

Figure 1: Type system for cryptographic protocols

Consider the type system presented in Figure 1, which is a simplified version of the type sytsem presented in the lecture (here we only consider symmetric encryption).

Problem 1: Typing for Cryptographic Protocols (30 Points)

a) **(15 Points)** Consider the following process:

$$(\nu v_1 : HH)\ (\nu v_2 : HL)\ (\nu k_1 : \mathrm{SymK}^{HH}[LH])\ (\nu k_2 : \mathrm{SymK}^{HH}[HL, HL, HH])\ \overline{b}\langle\{|[v_1, v_2, k_1]|\}^s_{k_2}\rangle.\mathbf{0}$$

Does it preserve secrecy? Does it preserve integrity? If yes, give a type derivation to prove your answer. If not, give a counter-example.

b) **(5 Points)** Consider the following process:

$$(\nu v_1 : HL)\ (\nu v_2 : LL)\ (\nu k_1 : \mathrm{SymK}^{HH}[HH, LH])\ b(x).\mathrm{case}\ x\ \mathrm{of}\ \{|[y, z]|\}^s_{k_1}\ \mathrm{in}\ (\nu k_2 : \mathbf{t})\ \overline{c}\langle\{|v_1, v_2, y, z|\}^s_{k_2}\rangle.\mathbf{0}$$

Give a concrete instantiation of the type $\mathbf{t}$ such that typing succeeds for the process. You do not have to write down the typing derivation.

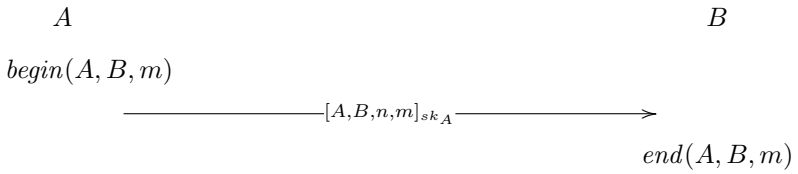
c) **(10 Points)** Show the incompleteness of the type system. Concretely, give an example of a process that preserves secrecy and integrity, but does not type check.

Problem 2: Protocol Analysis (20 Points)

Preliminaries In the following exercises we use the following notation:

- $\{m\}_{ek_A}$ is the encryption of the message m with the public encryption key of party A
- $[m]_{sk_A}$ is the signature of the message m with the private signing key of party A

a) **(10 Points)** Consider the following protocol, where m is a fresh message and n is a fresh nonce generated by A.

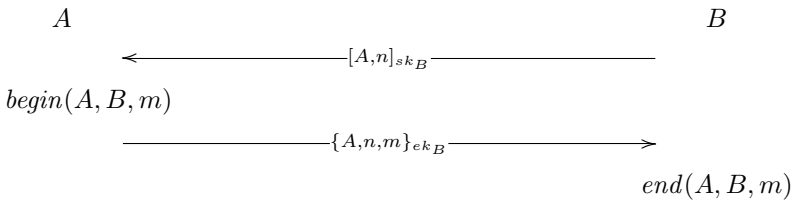


Does the protocol satisfy injective agreement? Does it satisfy non-injective agreement? If any of the two properties does not hold, describe the corresponding attack and propose a minimal modification to the protocol, such that it holds.

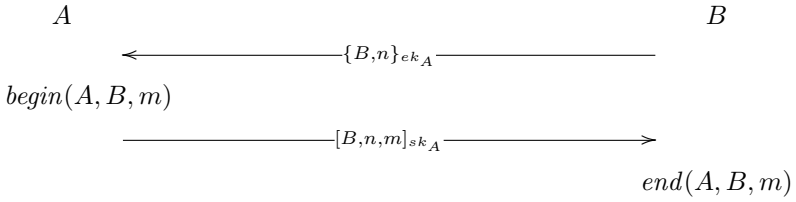
b) **(5 + 5 Points)** Consider the following two protocols, where m is a fresh message and n is a fresh nonce generated by B. For each of the two protocols either give an intuitive explanation, why it satisfies injective agreement or show an attack otherwise.

Assume that both A and B are willing to communicate with the attacker. Note however, that we do not consider *end* events containing the attacker's identifier for an attack against injective agreement.

(i)



(ii)



Problem 3: Information Flow (20 points)

Rules for typing expressions

$$\textsc{Int}\ \frac{}{n : L} \qquad \textsc{R-Val}\ \frac{\Gamma(x) = \tau var}{\Gamma \vdash x : \tau} \qquad \textsc{Op}\ \frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau \qquad \mathbf{op} \in \{+, -, >, <\}}{\Gamma \vdash e_1\ \mathbf{op}\ e_2 : \tau}$$

Rules for typing commands

$$\textsc{Skip}\ \frac{}{\Gamma \vdash \text{skip} : H\,cmd} \qquad \textsc{Assign}\ \frac{\Gamma(x) = \tau var \qquad \Gamma \vdash e : \tau}{\Gamma \vdash x := e : \tau\,cmd}$$

$$\textsc{If}\ \frac{\Gamma \vdash e : \tau \qquad \Gamma \vdash c_1 : \tau\,cmd \qquad \Gamma \vdash c_2 : \tau\,cmd}{\Gamma \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : \tau\,cmd} \qquad \textsc{While}\ \frac{\Gamma \vdash e : \tau \qquad \Gamma \vdash c : \tau\,cmd}{\Gamma \vdash \text{while } e \text{ do } c : \tau\,cmd}$$

$$\textsc{Compose}\ \frac{\Gamma \vdash c_1 : \tau\,cmd \qquad \Gamma \vdash c_2 : \tau\,cmd}{\Gamma \vdash c_1; c_2 : \tau\,cmd}$$

Subtyping rules

$$\textsc{Base}\ \frac{}{L \subseteq H} \qquad \textsc{Cmd}^{-}\ \frac{\tau' \subseteq \tau}{\tau\,cmd \subseteq \tau'\,cmd} \qquad \textsc{Reflex}\ \frac{}{\rho \subseteq \rho} \qquad \textsc{Trans}\ \frac{\rho_1 \subseteq \rho_2 \qquad \rho_2 \subseteq \rho_3}{\rho_1 \subseteq \rho_3}$$

$$\textsc{Subsump}\ \frac{\Gamma \vdash p : \rho_1 \qquad \rho_1 \subseteq \rho_2}{\Gamma \vdash p : \rho_2}$$

Figure 2: Type System for Non-interference.

a) **(10 Points)** Prove that the type system for non-interference (see Figure 2) is incomplete. A counterexample is sufficient.

b) **(10 Points)** For this exercise, use the type system for non-interference reported in Figure 2.

Assume that `l` : L and `h` : H. Try to infer a type for the following program. Show the type derivation. In case no type exists, briefly point out in the derivation where and why type-checking necessarily fails and argue whether the program is secure or not.

```
if l < 10 then
    while h > 0 do
        h := h - 1;
        l := 1
else
    skip
```

Problem 4: Static Analysis (30 points)

Assume the tiny goto language with the following instructions:

$$\mathsf{INIT}\ v, \mathsf{ADD}\ v, \mathsf{GOTO}\ i, \mathsf{GOTOifzx}\ i \qquad\qquad i \in \mathbb{N}, v \in \mathbb{Z}$$

You can think of programs in this language as simple imperative programs that can modify a variable x and otherwise can just alter the control flow by conditional and unconditional jumps. More precisely, the instruction $\mathsf{INIT}\ v$ initializes x with the value v, the instruction $\mathsf{ADD}\ v$ adds the value v to x and saves the result in x again, the instruction $\mathsf{GOTO}\ i$ jumps to the instruction at program counter i and finally $\mathsf{GOTOifzx}\ i$ jumps to the instruction at program counter i only if the value of variable x is 0.

More formally, we describe the state of a program by a configuration of the form $(pc, \mathsf{x} \mapsto v)$. Intuitively, such a configuration says that pc is the current program counter and variable x currently has value v. Next, we can define the semantics of the goto language in terms of a small-step relation. More precisely, $code \vdash (pc, \mathsf{x} \mapsto v) \to (pc', \mathsf{x} \mapsto v')$ means that given a program $code$ (which is just a list of instructions), the configuration changes from $(pc, \mathsf{x} \mapsto v)$ to $(pc', \mathsf{x} \mapsto v')$ within processing one instruction. Formally, the small-step relation is defined by the following inference rules:

$$\text{INIT}\ \frac{code[pc] = \mathsf{INIT}\ w}{code \vdash (pc, \mathsf{x} \mapsto v) \to (pc + 1, \mathsf{x} \mapsto w)} \qquad \text{ADD}\ \frac{code[pc] = \mathsf{ADD}\ w}{code \vdash (pc, \mathsf{x} \mapsto v) \to (pc + 1, \mathsf{x} \mapsto v + w)}$$

$$\text{GOTO}\ \frac{code[pc] = \mathsf{GOTO}\ i}{code \vdash (pc, \mathsf{x} \mapsto v) \to (i, \mathsf{x} \mapsto v)} \qquad \text{GOTOIFZX-FALSE}\ \frac{code[pc] = \mathsf{GOTOifzx}\ i \qquad v \neq 0}{code \vdash (pc, \mathsf{x} \mapsto v) \to (pc + 1, \mathsf{x} \mapsto v)}$$

$$\text{GOTOIFZX-TRUE}\ \frac{code[pc] = \mathsf{GOTOifzx}\ i \qquad v = 0}{code \vdash (pc, \mathsf{x} \mapsto v) \to (i, \mathsf{x} \mapsto v)}$$

a) **(6 Points)** Assume the following simple program:

```
1 INIT 1
2 GOTOifzx 5
3 ADD -1
4 GOTO 2
```

Starting in the configuration $(1, \mathsf{x} \mapsto 0)$ which configurations are reachable using the small step semantics? Name the rules (in the right order) that need to be applied to reach the corresponding configurations and give all intermediate configurations.

b) **(8 + 8 + 8 Points)** In the following, we will present different approaches to defining an abstract analysis for the presented language. As the versions of static analysis that were presented in the lecture, these approaches are based on Horn clause resolution. For each of the presented analysis approaches we define an abstraction function α that maps configurations into a state predicate (S) which models the abstract configuration. Analogously, we define a function h that defines the Horn clauses describing the abstract semantics.

Finally, we state soundness claims of the form

$$code \vdash c \to^* c' \implies \forall \Delta \geq \alpha(c).\ \exists \Delta' \geq \alpha(c').\ \Delta, h(code) \vdash \Delta'$$

which states that whenever configuration c executes $code$ and reaches configuration c' (within some number of steps) then it also holds that from any abstract configuration Δ that is at least as abstract as $\alpha(c)$ ($\Delta \geq \alpha(c)$) one can logically derive ($\vdash$) an abstract configuration Δ' that is at least as abstract as $\alpha(c')$ ($\Delta' \geq \alpha(c')$) using the Horn clauses $h(code)$.

Intuitively, an abstract configuration Δ is at least as abstract as an abstract configuration Δ' if for every fact (predicate application) in Δ' one can find one in Δ that is at least as abstract. Formally $\geq$ on abstract configurations is defined as follows:

$$\begin{aligned} \Delta \geq \Delta' :=&\quad \forall f' \in \Delta'.\ \exists f \in \Delta. f \geq_F f' \\ f \geq_F f' :=&\quad f = p(a_1, \ldots, a_n)\ \wedge\ f' = p(b_1, \ldots, b_n)\ \wedge\ \forall i \in \{1, \ldots, n\}.\ a_i \geq_A b_i \end{aligned}$$

Note that we also define what it means for a fact (predicate application) to be as abstract as another ($\geq_F$). A predicate application is at least as abstract as another if both are applications of the same predicate and all arguments of the first application are at least as abstract as the arguments of the second application at the respective position.

What it means for an argument value to be more abstract than another ($\geq_A$) depends on the argument domain A. If the arguments stem from the integer numbers ($A = \mathbb{Z}$) than a number is as abstract as itself, but there is no integer number that is more abstract than another. Some of the presented analysis approaches make use of the abstract domain $D = \{\top\} \cup \mathbb{Z}$ for arguments. Intuitively, $\top$ represents an arbitrary integer number and hence is more abstract than a concrete number. We formally define the relations $\geq_{\mathbb{Z}}$ and $\geq_D$:

$$\geq_{\mathbb{Z}} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{B} \qquad \geq_D : D \times D \to \mathbb{B}$$
$$x \geq_{\mathbb{Z}} y := \quad x = y \qquad a \geq_D b := \quad a = \top \ \vee \ a = b$$

Your task is to state for each of the presented analysis approaches if they satisfy their accompanying soundness claim.

- If yes, argue why the soundness claim holds and name (and explain) at least one point where the analysis looses precision. More concretely, give an example program *code* and initial configuration c such that that there is an abstract configuration Δ' that is derivable from $\alpha(c)$, but there is no concrete configuration c' that is reachable from c such that $\alpha(c') = \Delta'$. Provide also all reachable configurations c' explicitly, as well as $h(code)$.
- If not, give a counter example to the soundness claim. More precisely give a program *code* and initial configuration c, and an abstract configuration $\Delta \geq \alpha(c)$ such that there is a configuration c' that is reachable from c, but there is no $\Delta' \geq \alpha(c')$ that is derivable from Δ. Provide also c' and $h(code)$ explicitly and explain why no such Δ' is derivable.

(i) Predicates:

$$\{\mathsf{S}_i(a) \mid i \in \mathbb{N} \wedge a \in D\}$$

Abstraction of configurations:

$$\alpha_1((pc, \mathsf{x} \mapsto v)) = \{\mathsf{S}_{pc}(v)\}$$

Abstract rules:

$$\begin{aligned} h_1(code) = \\ &\{\mathsf{S}_{pc}(a) \Rightarrow \mathsf{S}_{pc+1}(v) \mid code[pc] = \mathsf{INIT}\ v\} \\ &\cup \{\mathsf{S}_{pc}(a) \Rightarrow \mathsf{S}_{pc+1}(\top) \mid code[pc] = \mathsf{ADD}\ v\} \\ &\cup \{\mathsf{S}_{pc}(a) \Rightarrow \mathsf{S}_i(a) \mid code[pc] = \mathsf{GOTO}\ i\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge (a = 0 \vee a = \top) \Rightarrow \mathsf{S}_i(a) \mid code[pc] = \mathsf{GOTOifzx}\ i\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge (a \neq 0 \vee a = \top) \Rightarrow \mathsf{S}_{pc+1}(a) \mid code[pc] = \mathsf{GOTOifzx}\ i\} \end{aligned}$$

Soundness claim:

$$code \vdash c \to^* c' \implies \forall \Delta \geq \alpha_1(c).\ \exists \Delta' \geq \alpha_1(c').\ \Delta, h_1(code) \vdash \Delta'$$

(ii) Predicates:

$$\{\mathsf{S}_i(a) \mid i \in \mathbb{N} \wedge a \in D\}$$

Abstraction of configurations:

$$\alpha_2((pc, \mathsf{x} \mapsto v)) = \{\mathsf{S}_{pc}(v)\}$$

Abstract rules:

$$\begin{aligned} h_2(code) = \\ &\{\mathsf{S}_{pc}(a) \Rightarrow \mathsf{S}_{pc+1}(v) \mid code[pc] = \mathsf{INIT}\ v\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge a \in \mathbb{Z} \Rightarrow \mathsf{S}_{pc+1}(a + v) \mid code[pc] = \mathsf{ADD}\ v\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge a = \top \Rightarrow \mathsf{S}_{pc+1}(\top) \mid code[pc] = \mathsf{ADD}\ v\} \\ &\cup \{\mathsf{S}_{pc}(a) \Rightarrow \mathsf{S}_i(a) \mid code[pc] = \mathsf{GOTO}\ i\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge a = 0 \Rightarrow \mathsf{S}_i(a) \mid code[pc] = \mathsf{GOTOifzx}\ i\} \\ &\cup \{\mathsf{S}_{pc}(a) \wedge a \neq 0 \Rightarrow \mathsf{S}_{pc+1}(a) \mid code[pc] = \mathsf{GOTOifzx}\ i\} \end{aligned}$$

Soundness claim:

$$\mathit{code} \vdash c \to^* c' \implies \forall \Delta \geq \alpha_2(c).\ \exists \Delta' \geq \alpha_2(c').\ \Delta, h_2(\mathit{code}) \vdash \Delta'$$

(iii) Predicates:

$$\{\mathsf{S}(v) \mid v \in \mathbb{Z}\}$$

Abstraction of configurations:

$$\alpha_3((pc, \mathsf{x} \mapsto v)) = \{\mathsf{S}(v)\}$$

Abstract rules:

$$\begin{aligned} h_3(\mathit{code}) = \\ &\{\mathsf{S}(v) \Rightarrow \mathsf{S}(w) \mid \mathsf{INIT}\ w \in \mathit{code}\} \\ &\cup \{\mathsf{S}(v) \Rightarrow \mathsf{S}(v+w) \mid \mathsf{ADD}\ w \in \mathit{code}\} \end{aligned}$$

Soundness claim:

$$\mathit{code} \vdash c \to^* c' \implies \forall \Delta \geq \alpha_3(c).\ \exists \Delta' \geq \alpha_3(c').\ \Delta, h_3(\mathit{code}) \vdash \Delta'$$

This is the last page.