PROVABLY CORRECT IMPLEMENTATION

Verifica del Software

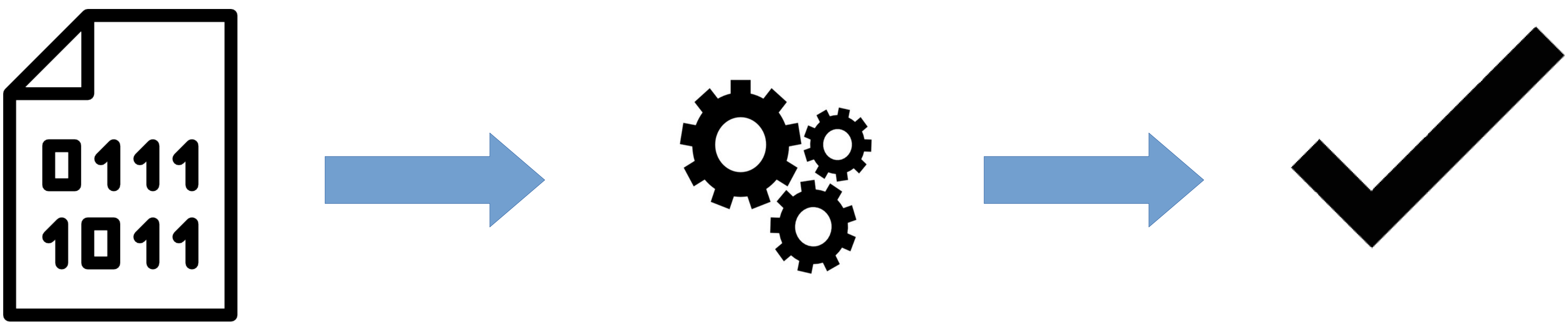
A.A. 2019-2020

Niccolo' Vettorello 1237291

INTRODUCTION

KEY IDEA

A *formal specification of the semantics* allows us to argue about the *correctness* of the implementation



INTRODUCTION

- **Abstract Machine (AM):** we'll give a formal specification of its set of instructions

- **Translation Functions:** replace statements in *While* with AM code sequences

INTRODUCTION

OUR STRATEGY

1) Define a formal semantics for AM
2) Define translation functions
3) Prove that if we translate a statement in While and execute it on AM, then the original meaning is preserved

ABSTRACT MACHINE

CONFIGURATIONS

$\langle c,e,s \rangle \in$ **Code** x **Stack** x **State**

- **Code:** c is a sequence of instructions
- **Stack:** $e \in \text{Stack} = (\mathbf{Z} \cup \mathbf{T})^*$
- **Storage:** we can imagine it as a state, i.e. $s \in \text{State}$

ABSTRACT MACHINE

SYNTAX

inst ::= PUSH-n | ADD | MULT | SUB |

TRUE | FALSE | EQ | LE | AND | NEG |

FETCH-x | STORE-x |

NOOP | BRANCH(c_1,c_2) | LOOP(c_1,c_2)

c ::= ε | inst:c

ABSTRACT MACHINE

OPERATIONAL SEMANTICS

<PUSH-n:c, e, s>	$\rhd$	<c, $\mathcal{N}$ |[n]|:e, s>
<ADD:c, z_1:z_2:e, s>	$\rhd$	<c, (z_1+z_2):e, s>
<SUB:c, z_1:z_2:e, s>	$\rhd$	<c, (z_1-z_2):e, s>
<MULT:c, z_1:z_2:e, s>	$\rhd$	<c, (z_1*z_2):e, s>
<TRUE:c, e, s>	$\rhd$	<c, tt:e, s>
<FALSE:c, e, s>	$\rhd$	<c, ff:e, s>

$z_1, z_2 \in \mathbf{Z}$

ABSTRACT MACHINE

OPERATIONAL SEMANTICS

$$<\mathrm{EQ}:c,\ z_1:z_2:e,\ s> \ \rhd\ <c,\ (z_1=z_2):e,\ s>$$

$$<\mathrm{LE}:c,\ z_1:z_2:e,\ s> \ \rhd\ <c,\ (z_1<=z_2):e,\ s>$$

$$<\mathrm{AND}:c,\ t_1:t_2:e,\ s> \ \rhd\ <c,\ (t_1 \wedge t_2):e,\ s>$$

$$<\mathrm{NEG}:c,\ t:e,\ s> \ \rhd\ <c,\ \neg t:e,\ s>$$

$$<\mathrm{FETCH\text{-}x}:c,\ e,\ s> \ \rhd\ <c,\ (s\ x):e,\ s>$$

$$<\mathrm{STORE\text{-}x}:c,\ z:e,\ s> \ \rhd\ <c,\ e,\ s[x \mapsto z]>$$

$z, z_1, z_2 \in \mathbf{Z}$

$t, t_1, t_2 \in \mathbf{T}$

ABSTRACT MACHINE

OPERATIONAL SEMANTICS

$$\langle BRANCH(c_1,c_2):c,\ t:e,\ s\rangle \ \rhd \begin{cases} \langle c_1:c,\ e,\ s\rangle & \text{if } t=tt \\ \langle c_2:c\ ,e,\ s\rangle & \text{if } t=ff \end{cases}$$

$$\langle LOOP(c_1,c_2):c,\ e,\ s\rangle \ \rhd$$
$$\langle c_1:BRANCH(c_2:LOOP(c_1,c_2),\ NOOP):c,\ e,\ s\rangle$$

$$\langle NOOP:c,\ e,\ s\rangle \rhd \quad \langle c,\ e,\ s\rangle$$

$c_1, c_2 \in$ **Code**

$t \in$ **T**

ABSTRACT MACHINE

Finite computation sequence

$$\gamma_0, \gamma_1, \ldots, \gamma_k$$

- $K \in \mathbf{N}$
- $\gamma_0 = \langle c, \varepsilon, s \rangle$
- $\gamma_i \rhd \gamma_{i+1} \; \forall \, i \in 0, \ldots, k\text{-}1$
- $\neg\exists \, \gamma \mid \gamma_k \rhd \gamma$

γ_k can be stuck or in final form, i.e. $\langle \varepsilon, e, s \rangle$

Infinite computation sequence

$$\gamma_0, \gamma_1, \ldots$$

- $\gamma_0 = \langle c, \varepsilon, s \rangle$
- $\gamma_i \rhd \gamma_{i+1} \; \forall \, i \in \mathbf{N}$

ABSTRACT MACHINE

Finite computation sequence: example

<PUSH-1:STORE-x, ε, s> ▻

<STORE-x, 1, s> ▻

<ε, ε, s[x↦1]>

Infinite computation sequence: example

<LOOP(TRUE,NOOP), ε, s> ▻

<TRUE:BRANCH(NOOP:LOOP(TRUE, NOOP), NOOP), ε, s> ▻

<BRANCH(NOOP:LOOP(TRUE, NOOP), NOOP), tt, s> ▻

. . .

ABSTRACT MACHINE

PROPERTIES

Composition Lemma

$$\langle c_1, e_1, s\rangle \rhd^k \langle c', e', s'\rangle \;\Rightarrow\; \langle c_1{:}c_2, e_1{:}e_2, s\rangle \rhd^k \langle c'{:}c_2, e'{:}e_2, s'\rangle$$

Proved by *structural induction* on the length of the computation sequence

Decomposition Lemma

$\langle c_1{:}c_2, e, s\rangle \rhd^k \langle \varepsilon, e'', s''\rangle \;\Rightarrow$

$\exists\, \gamma = \langle \varepsilon, e', s'\rangle,\; \exists\, k_1, k_2 \in \mathbf{N}$ such that

$\langle c_1, e, s\rangle \rhd^{k}{}_{1} \langle \varepsilon, e', s'\rangle \;\wedge\; \langle c_2, e', s'\rangle \rhd^{k}{}_{2} \langle \varepsilon, e'', s''\rangle$, with $k = k_1 + k_2$

PROPERTIES

Determinism

The operational semantics given for AM is *deterministic*, i.e.

$$\forall \gamma, \gamma', \gamma''$$

$$\gamma \rhd \gamma' \land \gamma \rhd \gamma'' \Rightarrow \gamma' = \gamma''$$

ABSTRACT MACHINE

EXECUTION FUNCTION *M*

M: Code $\rightarrow$ (State State)

NOTE: it's a final configuration

$$\mathcal{M}[[c]]\, s = \begin{cases} s' & \text{if } \langle c, \varepsilon, s\rangle \rhd^{*} \langle \varepsilon, e, s'\rangle \\ \underline{\text{undef}} & \text{otherwise} \end{cases}$$

1) The determinism of the semantics for AM entails the fact that *M* is well-defined

2) M is the meaning of a piece of code for AM

TRANSLATION FUNCTIONS

ARITHMETIC EXPRESSIONS

$$\mathcal{CA}: \text{Aexp} \rightarrow \text{Code}$$

$\mathcal{CA}|[n]| = \text{PUSH-n}$

$\mathcal{CA}|[x]| = \text{FETCH-x}$

$\mathcal{CA}|[a_1 + a_2]| = \mathcal{CA}|[a_2]|:\mathcal{CA}|[a_1]|:\text{ADD}$

$\mathcal{CA}|[a_1 - a_2]| = \mathcal{CA}|[a_2]|:\mathcal{CA}|[a_1]|:\text{SUB}$

$\mathcal{CA}|[a_1 * a_2]| = \mathcal{CA}|[a_2]|:\mathcal{CA}|[a_1]|:\text{MULT}$

TRANSLATION FUNCTIONS

BOOLEAN EXPRESSIONS

CB: Bexp → Code

CB |[true]| = TRUE

CB |[false]| = FALSE

CB |[a_1 = a_2]| = CA|[a_2]|:CA|[a_1]|:EQ

CB |[a_1 <= a_2]| = CA|[a_2]|:CA|[a_1]|:LE

CB |[¬b]| = CB|[b]|:NEG

CB |[$b_1 \wedge b_2$]| = CB |[b_2]|:CB |[b_1]|:AND

PROPERTIES OF *CA* AND *CB*

1) *CA* and *CB* definition is compositional

2) *CA* and *CB* definition preserves the order of operands

TRANSLATION FUNCTIONS

STATEMENTS

$$\mathcal{CS} : \text{Stm} \to \text{Code}$$

$\mathcal{CS}|[x:=a]| = \mathcal{CA}|[a]|:\text{STORE-x}$

$\mathcal{CS}|[\text{skip}]| = \text{NOOP}$

$\mathcal{CS}|[S_1;S_2]| = \mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|$

$\mathcal{CS}|[\text{if b then } S_1 \text{ else } S_2]| = \mathcal{CB}|[b]|:\text{BRANCH}(\mathcal{CS}|[S_1]|,|[S_2]|)$

$\mathcal{CS}|[\text{while b do S}]| = \text{LOOP}(\mathcal{CB}|[b]|, \mathcal{CS}|[S]|)$

This definition is *compositional*

SEMANTIC FUNCTION S_{AM}

We can now express the meaning of a statement in While by means of the semantic function S_{AM} defined as:

$$S_{AM}: \text{Stm} \rightarrow (\text{State} \hookrightarrow \text{State})$$

$$S_{AM} = \mathcal{M} \circ \mathcal{CS}$$

Represents the meaning of the AM code

Translates the statement Into AM code

ARITHMETIC EXPRESSIONS

We want to show that:

$$\forall\, a \in \mathrm{Aexp} \qquad <\mathcal{CA}|[a]|,\ \varepsilon,\ s> \rhd^{*} <\varepsilon,\ \mathcal{A}|[a]|s,\ s>$$

Furthermore, every intermediate configuration will have a non-empty stack.

The proof is based on *structural induction on a*

ARITHMETIC EXPRESSIONS: PROOF

Case: *n*

1) *CA*|[n]| = PUSH-n — by definition of *CA*
2) <PUSH-n, ε, s> ⊳ <ε, *N*|[n]|, s> — semantics of AM
3) *A*|[n]|s = *N*|[n]| — by definition of *A*

It's easily observed that every configuration has a non-empty stack

Case: *x*

1) *CA*|[x]| = FETCH-x — by definition of *CA*
2) <FETCH-x, ε, s> ⊳ <ε, (s x), s> — semantics of AM
3) *A*|[x]|s = s x — by definition of *A*

CORRECTNESS

ARITHMETIC EXPRESSIONS: PROOF

Case: $a_1 + a_2$

1) $\mathcal{CA}|[a_1 + a_2]| = \mathcal{CA}|[a_2]|:\mathcal{CA}|[a_1]|:\text{ADD}$ by definition of *CA*

2) $\langle \mathcal{CA}|[a_1]|, \varepsilon, s\rangle \rhd^* \langle \varepsilon, \mathcal{A}|[a_1]|s, s\rangle$ by inductive hypothesis

3) $\langle \mathcal{CA}|[a_2]|, \varepsilon, s\rangle \rhd^* \langle \varepsilon, \mathcal{A}|[a_2]|s, s\rangle$ by inductive hypothesis

Note that in 2) and 3) every intermediate configuration has a non-empty stack

CORRECTNESS

ARITHMETIC EXPRESSIONS: PROOF

4) $<\mathcal{CA}|[a_2]|:\mathcal{CA}|[a_1]|:\text{ADD}, \varepsilon, s> \rhd^* <\mathcal{CA}|[a_1]|:\text{ADD}, \mathcal{A}|[a_2]|s, s>$
by Composition Lemma

5) $<\mathcal{CA}|[a_1]|:\text{ADD}, \mathcal{A}|[a_2]|s, s> \rhd^* <\text{ADD}, \mathcal{A}|[a_1]|s:\mathcal{A}|[a_2]|s, s>$
by Composition Lemma

6) $<\text{ADD}, \mathcal{A}|[a_1]|s:\mathcal{A}|[a_2]|s, s> \rhd <\varepsilon, \mathcal{A}|[a_1]|s+\mathcal{A}|[a_2]|s, s>$
by semantics of AM

7) $\mathcal{A}|[a_1]|s+\mathcal{A}|[a_2]|s = \mathcal{A}|[a_1 + a_2]|s$ by definition of A

ARITHMETIC EXPRESSIONS: PROOF

Case: $a_1 - a_2$

Analogous

Case: $a_1 * a_2$

Analogous

CORRECTNESS

BOOLEAN EXPRESSIONS

In a similar way, we can prove that

$$\forall\, b \in \mathrm{Bexp} \qquad <\mathcal{CB}|[b]|, \varepsilon, s> \rhd^{*} <\varepsilon, \mathcal{B}|[B]|s, s>$$

And every intermediate configuration will have a non-empty stack.

CORRECTNESS

STATEMENTS

We want to prove that

$$\forall\, S \in Stm \qquad S_{AM}|[S]| = S_{ns}|[S]|$$

That is:

- If the execution of S terminates in the context of natural semantics then it terminates on AM too, and vice versa;
- If the execution of S loops in the context of natural semantics then it loops on AM too, and vice versa.

STATEMENTS

Lemma A: $\forall$ S $\in$ Stm, $\forall$ s, s’ $\in$ State

$$\langle S, s\rangle \to s' \;\Rightarrow\; \langle CS|[S]|, \varepsilon, s\rangle \rhd^{*} \langle \varepsilon, \varepsilon, s'\rangle$$

Lemma B: $\forall$ S $\in$ Stm, $\forall$ s, s’ $\in$ State

$$\langle CS|[S]|, \varepsilon, s\rangle \rhd^{k} \langle \varepsilon, e, s'\rangle \;\Rightarrow\; (\langle S, s\rangle \to s' \;\wedge\; e=\varepsilon)$$

CORRECTNESS

$[\text{ass}_{\text{ns}}]$	$\langle x := a,\ s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$
$[\text{skip}_{\text{ns}}]$	$\langle \texttt{skip},\ s\rangle \to s$
$[\text{comp}_{\text{ns}}]$	$\dfrac{\langle S_1,\ s\rangle \to s',\ \langle S_2,\ s'\rangle \to s''}{\langle S_1;S_2,\ s\rangle \to s''}$
$[\text{if}^{\text{tt}}_{\text{ns}}]$	$\dfrac{\langle S_1,\ s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'}$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{\text{ff}}_{\text{ns}}]$	$\dfrac{\langle S_2,\ s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'}$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}^{\text{tt}}_{\text{ns}}]$	$\dfrac{\langle S,\ s\rangle \to s',\ \langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \to s''}{\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s''}$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{while}^{\text{ff}}_{\text{ns}}]$	$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$

Table 2.1 Natural semantics for **While**

LEMMA A: PROOF

The proof proceeds by *induction on the shape of the derivation tree* for ‹S, s› → s’

Case: *[ass$_{ns}$]*

We have ‹x:=a, s› → s[x↦A|[a]|s] as hypothesis;

1) *CS*|[x:=a]| = *CA*|[a]|:STORE-x — by definition of *CS*

2) <*CA*|[a]|, ε, s> ▻* <ε, *A*|[a]|s, s> — by the correctness of *CA*

3) <*CA*|[a]|:STORE-x, ε, s> ▻*<STORE-x, *A*|[a]|s, s> — by Composition Lemma

4) <STORE-x, *A*|[a]|s,s> ▻ <ε, ε, s[x↦*A*|[a]|s]> — by semantics of AM

LEMMA A: PROOF

Case: *[$skip_{ns}$]*

We have ‹skip, s› → s as hypothesis;

1) *CS* |[skip]| = NOOP by definition of *CS*

2) <NOOP, ε, s> ▻ <ε, ε, s> by semantics of AM

CORRECTNESS

LEMMA A: PROOF

Case: *[comp$_{ns}$]*

We have $\langle S_1;S_2, s\rangle \to s''$ as hypothesis. By induction hypothesis we get:

I. $<\mathcal{CS}|[S_1]|, \varepsilon, s> \rhd^* <\varepsilon, \varepsilon, s'>$
II. $<\mathcal{CS}|[S_2]|, \varepsilon, s'> \rhd^* <\varepsilon, \varepsilon, s''>$

It follows that:

1) $\mathcal{CS}|[S_1;S_2]| = \mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|$ by definition of CS

2) $<\mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|, \varepsilon, s> \rhd^* <\mathcal{CS}|[S_2]|, \varepsilon, s'>$ by Composition Lemma and IH 1

3) $<\mathcal{CS}|[S_2]|, \varepsilon, s'> \rhd^* <\varepsilon, \varepsilon, s''>$ by IH 2

CORRECTNESS

LEMMA A: PROOF

Case: $[if_{ns}]^{tt}$

We assume ‹if b then S_1 else S_2, s› → s' and $\mathcal{B}$|[b]|s = tt . By induction hypothesis we get:

$$<\mathcal{CS}|[S_1]|, \varepsilon, s> \rhd^* <\varepsilon, \varepsilon, s'>$$

It follows that:

1) $\mathcal{CS}$|[if b then S_1 else S_2]| = $\mathcal{CB}$|[b]|:BRANCH($\mathcal{CS}$|[S_1]|,$\mathcal{CS}$|[S_2]|) by definition of $\mathcal{CS}$

2) <$\mathcal{CB}$|[b]|:BRANCH($\mathcal{CS}$|[S_1]|,$\mathcal{CS}$|[S_2]|), ε, s> $\rhd^*$ <BRANCH($\mathcal{CS}$|[S_1]|,$\mathcal{CS}$|[S_2]|), $\mathcal{B}$|[b]|s, s>
by Composition Lemma and correctness of $\mathcal{CB}$

3) <BRANCH($\mathcal{CS}$|[S_1]|,$\mathcal{CS}$|[S_2]|), $\mathcal{B}$|[b]|s, s> $\rhd$ <$\mathcal{CS}$|[S_1]|,ε, s> $\rhd^*$ <ε, ε, s'>
by semantics of AM, $\mathcal{B}$|[b]|s = tt and IH

LEMMA A: PROOF

Case: $[\mathit{if}_{ns}]^{\mathit{ff}}$

Analogous.

CORRECTNESS

LEMMA A: PROOF

Case: *[while$_{ns}$]^{tt}*

We assume ‹while b do S, s› → s'' and $\mathcal{B}$|[b]|s = tt . By induction hypothesis we get:

I. <$\mathcal{CS}$|[S]|, ε, s> ▻* <ε, ε, s'>
II. <LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|), ε, s'> ▻* <ε, ε, s''>

It follows that:

1) $\mathcal{CS}$|[while b do S]| = LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|) — by definition of *CS*

2) <LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|), ε, s> ▻
<$\mathcal{CB}$|[b]|:BRANCH($\mathcal{CS}$|[s]|:LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|),NOOP), ε, s> — by semantics of AM

3) <$\mathcal{CB}$|[b]|:BRANCH($\mathcal{CS}$|[s]|:LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|),NOOP), ε, s> ▻*
<BRANCH($\mathcal{CS}$|[s]|:LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|),NOOP), $\mathcal{B}$|[b]|s, s> — by Composition Lemma and correctness of *CB*

LEMMA A: PROOF

4) <BRANCH(*CS*|[s]|:LOOP(*CB*|[b]|, *CS*|[S]|),NOOP), *B*|[b]|s, s> $\rhd$
<*CS*|[s]|:LOOP(*CB*|[b]|, *CS*|[S]|), ε, s>
by semantics of AM, B|[b]|s = tt

5) <*CS*|[s]|:LOOP(*CB*|[b]|, *CS*|[S]|), ε, s> $\rhd^*$
<LOOP(*CB*|[b]|, *CS*|[S]|), ε, s'> $\rhd^*$
<ε, ε, s''>
by IH 1, Composition Lemma, IH 2

CORRECTNESS

LEMMA A: PROOF

Case: *$[while_{ns}]^{ff}$*

We assume ‹while b do S, s› → s and $\mathcal{B}$|[b]|s = ff .
We have:

1) $\mathcal{CS}$|[while b do S]| = LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|)
   by definition of *CS*

2) <LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|), ε, s> ▻
   <$\mathcal{CB}$|[b]|:BRANCH($\mathcal{CS}$|[s]|:LOOP($\mathcal{CB}$|[b]|, $\mathcal{CS}$|[S]|),NOOP), ε, s>
   by semantics of AM

CORRECTNESS

LEMMA A: PROOF

3) $\langle \mathit{CB}|[b]|:\mathrm{BRANCH}(\mathit{CS}|[s]|:\mathrm{LOOP}(\mathit{CB}|[b]|,\ \mathit{CS}|[S]|),\mathrm{NOOP}),\ \varepsilon,\ s\rangle \rhd^{*}$

$\langle \mathrm{BRANCH}(\mathit{CS}|[s]|:\mathrm{LOOP}(\mathit{CB}|[b]|,\ \mathit{CS}|[S]|),\mathrm{NOOP}),\ \mathit{B}|[b]|s,\ s\rangle \rhd$

$\langle \mathrm{NOOP},\ \varepsilon,\ s\rangle \rhd$

$\langle \varepsilon,\ \varepsilon,\ s\rangle$

by correctness of CB, Composition Lemma, $B|[b]|s = ff$, semantics of AM

□

LEMMA B: PROOF

The proof proceeds by i*nduction on the length k of the computation sequence*: for k = 0 the result holds vacuously, since *CS*|[S]| = ε is not possible.

We assume it holds for k <= k_0 and prove that it holds for k = k_0+1.

Case: *skip*

1) *CS*|[skip]| = NOOP by definition of *CS*

2) <NOOP, ε, s> ▻ <ε, ε, s> by semantics of AM

LEMMA B: PROOF

Case: *x:=a*

We assume <$\mathcal{CA}$|[a]|:STORE-x, ε, s> $\triangleright^{k_0+1}$ <ε, e, s'>. By Decomposition Lemma exists <ε, e'', s''> such that:

I. <$\mathcal{CA}$|[a]|, ε, s> $\triangleright^{k_1}$ <ε, e'', s''>

II. <STORE-x, e'', s''> $\triangleright^{k_2}$ <ε, e, s'>

And $k_1 + k_2 = k_0 + 1$.

We have that:

1) e'' = $\mathcal{A}$|[a]|s and s = s'' by correctness of CA and determinism

2) s' = s[x↦$\mathcal{A}$|[a]|s] ∧ e = ε by semantics of AM

CORRECTNESS

LEMMA B: PROOF

Case: $S_1;S_2$

We assume $<CS|[S_1]|:CS|[S_2]|, \varepsilon, s> \rhd^{k_0+1} <\varepsilon, e, s'>$. By Decomposition Lemma exists $<\varepsilon, e'', s''>$ such that:

I. $<CS|[S_1]|, \varepsilon, s> \rhd^{k_1} <\varepsilon, e'', s''>$

II. $<CS|[S_2]|, e'', s''> \rhd^{k_2} <\varepsilon, e, s'>$

And $k_1 + k_2 = k_0 + 1$.

IH tells us that

a) $\langle S_1, s\rangle \to s'' \wedge e'' = \varepsilon$

b) $\langle S_2, s''\rangle \to s' \wedge e = \varepsilon$

The rule [$comp_{ns}$] gives the desired result.

CORRECTNESS

LEMMA B: PROOF

Case: *if b then S_1 else S_2*

We assume <*CB*|[b]|:BRANCH(*CS*|[S_1]|,*CS*|[S_2]|), ε, s> $\rhd^{k_0+1}$ <ε, e, s'>. By Decomposition Lemma exists <ε, e'', s''> such that:

I. <*CB*|[b]|, ε, s> $\rhd^{k_1}$ <ε, e'', s''>

II. <BRANCH(*CS*|[S_1]|,*CS*|[S_2]|), e'', s''> $\rhd^{k_2}$ <ε, e, s'>

And $k_1 + k_2 = k_0 + 1$.

We note that, by the correctness of *CB* and determinism,

e'' = *B*|[b]|s and s'' = s.

LEMMA B: PROOF

Now:

a) $\mathcal{B}[[b]]s = tt$

$<\mathcal{CS}|[S_1]|, \varepsilon, s> \rhd^{k_2 - 1} <\varepsilon, e, s'>$
By IH we know that $e = \varepsilon$ and that $\langle S_1, s\rangle \rightarrow s'$. We conclude by using the rule $[if_{ns}]^{tt}$

b) $\mathcal{B}[[b]]s = tt$

Analogous.

CORRECTNESS

LEMMA B: PROOF

Case: *while b do S*

We assume <LOOP(*CB*|[b]|,*CS*|[S]|), ε, s> $\rhd^{k_0+1}$ <ε, e, s'>.

By the semantics of AM, we have

<LOOP(*CB*|[b]|,*CS*|[S]|), ε, s> $\rhd$
<*CB*|[b]|:BRANCH(*CS*|[s]|:LOOP(*CB*|[b]|, *CS*|[S]|),NOOP), ε, s> $\rhd^{k_0}$
<ε, e, s'>

LEMMA B: PROOF

By Decomposition Lemma exists <ε, e'', s''> such that:

1. $<CB|[b]|, \varepsilon, s> \rhd^{k}{}_{1} <\varepsilon, e'', s''>$
2. $<\text{BRANCH}(CS|[s]|:\text{LOOP}(CB|[b]|, CS|[S]|),\text{NOOP}), e'', s''> \rhd^{k}{}_{2}$
   $<\varepsilon, e, s'>$

And $k_1 + k_2 = k_0 + 1$.

We note that, by the correctness of *CB* and determinism,

$e'' = B|[b]|s$ and $s'' = s$.

CORRECTNESS

LEMMA B: PROOF

1) *B|[b]|s = tt*

<*CS*|[S]|:LOOP(*CB*|[b]|, *CS*|[S]|), ε, s> $\rhd^{k}_{2}{}^{-1}$ <ε, e, s'> by the semantics of AM

Again by Decomposition Lemma, there exists <ε, e''', s'''> such that:

a) <*CS*[S]|, ε, s> $\rhd^{k}_{3}$ <ε, e''', s'''>
b) <LOOP(*CB*|[b]|, *CS*|[S]|), e''', s'''> $\rhd^{k}_{4}$ <ε, e, s'>

And k3 + k4 = k2 - 1

LEMMA B: PROOF

We can apply IH on a) to obtain:

$$\langle S, s\rangle \to s''' \wedge e''' = \varepsilon$$

Again by IH on b) we get:

$$\langle \text{while } b \text{ do } S, s'''\rangle \to s \wedge e = \varepsilon$$

Then we conclude by using the rule $[\mathit{while}_{ns}]^{tt}$

LEMMA B: PROOF

2) $\mathcal{B}[[b]]s = ff$

We get:

$\langle NOOP, \varepsilon, s\rangle \rhd \langle \varepsilon, \varepsilon, s\rangle$ by semantics of AM

It's easy to see that e = ε and s = s'. We conclude using the rule $[while_{ns}]^{ff}$

$\square$

AN ALTERNATIVE PROOF TECHNIQUE

We want to prove that s.o.s. semantics and AM semantics are equivalent, i.e.

$\forall$ S $\in$ Stm $\qquad S_{AM}|[S]| = S_{sos}|[S]|$

The proof for this theorem is based on the idea that the two semantics proceed in *lockstep*: one is able to find corresponding sequences that produce similar changes in configurations.

The concept of similarity between configurations is captured by the *bisimulation relation*, defined as:

$\forall$ S $\in$ Stm

$$s \approx \langle \varepsilon, \varepsilon, s \rangle$$

$$\langle S, s \rangle \approx \langle \mathcal{CS}|[S]|, \varepsilon, s \rangle$$

AN ALTERNATIVE PROOF TECHNIQUE

$[\text{ass}_{\text{sos}}]$	$\langle x := a,\ s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$
$[\text{skip}_{\text{sos}}]$	$\langle \texttt{skip},\ s\rangle \Rightarrow s$
$[\text{comp}^1_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_1';S_2,\ s'\rangle}$
$[\text{comp}^2_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow s'}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_2,\ s'\rangle}$
$[\text{if}^{\text{tt}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_1,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{\text{ff}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_2,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}_{\text{sos}}]$	$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \Rightarrow$ $\langle \texttt{if}\ b\ \texttt{then}\ (S;\ \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip},\ s\rangle$

Table 2.2 Structural operational semantics for **While**

AN ALTERNATIVE PROOF TECHNIQUE

Lemma A:

$$\gamma_{sos} \approx \gamma_{am} \wedge \gamma_{sos} \Rightarrow \gamma'_{sos}$$

then

$$\exists\, \gamma'_{am} \text{ such that } \gamma_{am} \rhd^{+} \gamma'_{am} \wedge \gamma'_{sos} \approx \gamma'_{am}$$

$\rhd^{+}$

is defined as

$\rhd \lozenge \rhd^{*}$

Furthermore, if $\langle S, s\rangle \Rightarrow^* s'$ then $<\mathcal{CS}|[S]|, \varepsilon, s> \rhd^* <\varepsilon, \varepsilon, s'>$.

Lemma A states that whenever one step of structural operational semantics changes the configuration, there exists a sequence of steps in AM semantics that will produce a similar change.

AN ALTERNATIVE PROOF TECHNIQUE

Lemma B:

$$\gamma_{sos} \approx \gamma^1{}_{am} \;\wedge$$

$$\gamma^1{}_{am} \rhd \gamma^2{}_{am} \rhd \ldots \rhd \gamma^k{}_{am} \text{ with } k>1 \text{and only } \gamma^k{}_{am} \text{ and } \gamma^1{}_{am} \text{ have empty stack}$$

then

$$\exists\, \gamma'_{sos} \text{ such that } \gamma_{sos} \Rightarrow \gamma'_{sos} \wedge \gamma'_{sos} \approx \gamma^k{}_{am}$$

Furthermore, if $<\mathcal{CS}|[S]|, \varepsilon, s> \rhd^* <\varepsilon, \varepsilon, s'>$ then $\langle S, s\rangle \Rightarrow^* s'$.

Lemma B states that whenever AM makes a sequence of steps from one configuration with empty stack to another configuration with empty stack, the structural operational semantics can make a similar change.

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

The proof proceeds by *induction on the shape of the derivation tree* for the single s.o.s. step.

By hypothesis we know that s.o.s. makes a step. This rules out the possibility that $\gamma_{sos} = s$. In fact γ_{sos} must be in the form ‹S,s>, so we prove the validity of the Lemma for every type of statement.

Case: *skip*

1) ‹skip,s› ≈ <NOOP, ε, s> ∧ ‹skip,s› ⇒ s   our hypothesis
2) <NOOP, ε, s> ▻ <ε, ε, s> ≈ s   by semantics of AM

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

Case: *x:=a*

1) ‹x:=a,s› ≈ <*CA*|[a]|:STORE-x, ε, s> ∧ ‹x:=a,s› ⇒ s[x↦*A*|[a]|s]
   our hypothesis

2) <*CA*|[a]|, ε, s> ▻* <ε, *A*|[a]|s, s>

   by correctness of CA

3) <*CA*|[a]|:STORE-x, ε, s>▻* <STORE-x, *A*|[a]|s, s> ▻
   <ε, ε, s[x↦*A*|[a]|s]> ≈ s[x↦*A*|[a]|s]
   by Composition Lemma and semantics of AM

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

Case: $S_1;S_2$

$\langle S_1;S_2, s\rangle \approx <\mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|, \varepsilon, s>$ our hypothesis

1) The rule used is [$comp^1_{sos}$], i.e. our hypothesis

$\langle S_1;S_2, s\rangle \Rightarrow \langle S'_1;S_2, s'\rangle$
because
$\langle S_1, s\rangle \Rightarrow \langle S'_1, s'\rangle$

a) IH tells us that $<\mathcal{CS}|[S_1]|, \varepsilon, s> \rhd^+ <\mathcal{CS}|[S'_1]|, \varepsilon, s'>$

b) $<\mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|, \varepsilon, s> \rhd^+ <\mathcal{CS}|[S'_1]|:\mathcal{CS}|[S_2]|, \varepsilon, s'> \approx \langle S'_1;S_2, s'\rangle$ by Composition Lemma

LEMMA A: PROOF

2) The rule used is [$comp^2_{sos}$], i.e. our hypothesis

$\langle S_1;S_2, s\rangle \Rightarrow \langle S_2, s'\rangle$

because

$\langle S_1,s\rangle \Rightarrow s'$

a) IH tells us that $<\mathcal{CS}|[S_1]|, \varepsilon, s> \rhd^+ <\varepsilon, \varepsilon, s'>$

b) $<\mathcal{CS}|[S_1]|:\mathcal{CS}|[S_2]|, \varepsilon, s> \rhd^+ <\mathcal{CS}|[S_2]|, \varepsilon, s'> \approx \langle S_2, s'\rangle$

by Composition Lemma

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

Case: *if b then S_1 else S_2*

$\langle$if b then S_1 else S_2,s$\rangle$ $\approx$ $<\mathcal{CB}|[b]|$:BRANCH($\mathcal{CS}|[S_1]|,\mathcal{CS}|[S_2]|$), ε, s$>$ our hypothesis

We note that:

$<\mathcal{CB}|[b]|$:BRANCH($\mathcal{CS}|[S_1]|,\mathcal{CS}|[S_2]|$), ε, s$>$ $\rhd^*$ $<$BRANCH($\mathcal{CS}|[S_1]|,\mathcal{CS}|[S_2]|$), $\mathcal{B}|[b]|$s, s$>$

by correctness of *CB* and Composition Lemma

1) The rule used is $[if_{sos}]^{tt}$: we have $\mathcal{B}|[b]|$s = tt and $\langle$if b then S_1 else S_2, s$\rangle \Rightarrow \langle S_1$, s$\rangle$

$<$BRANCH($\mathcal{CS}|[S_1]|,\mathcal{CS}|[S_2]|$), $\mathcal{B}|[b]|$s, s$>$ $\rhd$ $<\mathcal{CS}|[S_1]|$, ε, s$>$ $\approx$ $\langle S_1$, s$\rangle$

by semantics of AM

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

2) The rule used is $[if_{sos}]^{ff}$ and we have $\mathcal{B}[\![b]\!]s = ff$ and

$\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$

Analogous.

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

Case: *while b do S*

‹while b do S, s› ≈ <LOOP(*CB*|[b]|,*CS*|[S]|), ε, s>

∧

‹while b do S, s› ⇒ ‹if b then (S; while b do S) else skip,s›

It’s easy to see that:

<LOOP(*CB*|[b]|,*CS*|[S]|), ε, s> ▻ by semantics of AM

<*CB*|[b]|:BRANCH(*CS*|[s]|:LOOP(*CB*|[b]|, *CS*|[S]|),NOOP), ε, s>

which is in bisimulation relation with:

‹if b then (S; while b do S) else skip,s›

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

if $\langle S,s\rangle \Rightarrow^* s'$

then

$<\mathcal{CS}|[S]|, \varepsilon, s> \rhd^* <\varepsilon, \varepsilon, s'>$

We proceed by *induction on the length k* of $\langle S,s\rangle \Rightarrow^k s'$

Case: $k = 0$

This means that $\langle S,s\rangle \Rightarrow^0 s'$, so the thesis is vacuously true.

AN ALTERNATIVE PROOF TECHNIQUE

LEMMA A: PROOF

Case: $k = k_0 + 1$

1) $\langle S,s\rangle \Rightarrow \gamma'_{sos} \Rightarrow^k{}_0\ s'$

   by hypothesis

2) $\exists\ \gamma'_{am}$ such that $<\mathcal{CS}|[S]|, \varepsilon, s> \rhd^* \gamma'_{am} \wedge \gamma'_{sos} \approx \gamma'_{am}$

   by the first part of Lemma A

3) $\exists\ \gamma''_{am}$ such that $\gamma'_{sos} \approx \gamma''_{am}$ and $\gamma''_{am} \rhd^* <\varepsilon, \varepsilon, s'>$

   by IH

4) $\gamma'_{am} = \gamma''_{am}$

   by definition of bisimulation

□