Semantics with Applications
Natural Semantics

Pablo López

University of Málaga

November 14, 2023

Outline

Outline

- Introduction
- Natural Semantics
- Properties of the Natural Semantics

Abstract Syntax of WHILE

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2 \\
b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & x := a \mid \texttt{skip} \mid S_1 \,;\, S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \\
 & \mid & \texttt{while } b \texttt{ do } S
\end{array}
$$

We covered the semantics of arithmetic ($\mathbf{Aexp}$) and Boolean ($\mathbf{Bexp}$) expressions.
We still have to cover the semantics of statements ($\mathbf{Stm}$).

Dealing with Change

The purpose of statements in `WHILE` is to change the state:

- the semantics of $\mathbf{Aexp}$ and $\mathbf{Bexp}$ only *inspect* the state
  - evaluation is side-effect free
- the semantics of $\mathbf{Stm}$ can *modify* the state
  - execution changes the state

Operational Semantics and State Change

- the execution of a program changes its state
- operational semantics are concerned with **how** to execute programs; not merely with the final results
- we are interested in how the states are modified **during** the execution
- therefore, operational semantics must model **state change**

Two Styles of Operational Semantics

What states are relevant?

- **Natural Semantics** (*big-step* semantics) describe how the *overall* results are obtained from *initial* to *final* state
- **Structural Operational Semantics** (*small-step* semantics) describe how the individual steps change the states (initial, intermediate, and final)

We model both operational semantics by **transition systems**.

Transition System

A **transition system** is a tuple $(\Gamma, T, \triangleright)$ where:

- Γ is a set of **configurations**
- T a set of **terminal configurations** $T \subseteq \Gamma$
- $\triangleright$ is a **transition relation** $\triangleright \subseteq \Gamma \times \Gamma$

Quiz

Recall that the transition relation $\triangleright$ is defined as $\triangleright \subseteq \Gamma \times \Gamma$.

Quiz

Recall that the transition relation $\triangleright$ is defined as $\triangleright \subseteq \Gamma \times \Gamma$.
An alternative definition of $\triangleright$ is $\triangleright \subseteq (\Gamma \setminus T) \to \Gamma$.

Quiz

Recall that the transition relation $\triangleright$ is defined as $\triangleright \subseteq \Gamma \times \Gamma$.
An alternative definition of $\triangleright$ is $\triangleright \subseteq (\Gamma \setminus T) \to \Gamma$.
What's the difference?

Quiz

Recall that the transition relation $\triangleright$ is defined as $\triangleright \subseteq \Gamma \times \Gamma$.
An alternative definition of $\triangleright$ is $\triangleright \subseteq (\Gamma \setminus T) \rightarrow \Gamma$.
What's the difference?

Configurations for `WHILE`

We define two types of **configurations**:

- $\langle S, s \rangle$ statement S is to be executed from the state s, and
- s terminal or final state

A configuration of the latter form is a **terminal configuration**.

Configurations for `WHILE`

We define two types of **configurations**:

- $\langle S, s\rangle$ statement S is to be executed from the state s, and
- s terminal or final state

A configuration of the latter form is a **terminal configuration**.
The **natural** and **structural operational** semantics:

- use the same sets of configurations, Γ and T
- differ in the definition of the transition relation $\triangleright$.

Since `WHILE` is deterministic, we shall replace $\triangleright$ by $\rightarrow$

Outline

Transition System for Natural Semantics

The Natural Semantics of `WHILE` is defined by a transition system $(\Gamma, T, \rightarrow)$ where:

$$
\begin{array}{rcl}
\Gamma & = & \{\langle S, s\rangle \mid S \in \mathbf{Stm},\ s \in \mathbf{State}\} \cup \mathbf{State} \\
T & = & \mathbf{State} \\
\rightarrow & \subseteq & \Gamma \times \mathbf{State}
\end{array}
$$

Fundamentals of Natural Semantics

- We are concerned with the **initial** and **final** states
- The transition relation $\rightarrow$ specifies the relationship between the initial and the final states for each statement of `WHILE`
- The transition or **judgement**

$$\langle S, s\rangle \rightarrow s'$$

  means that the execution of statement S from the initial state s will **terminate**, yielding the final state s'.

But how do we know it?

- Given a judgement $\langle S, s\rangle \to s'$, how do we know if it holds?
- For example, how do we know that

  $$\langle \mathtt{y} := \mathtt{y} - 1; \mathtt{skip}; \mathtt{x} := \mathtt{x} + 1, [\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 4]\rangle \to [\mathtt{x} \mapsto 4, \mathtt{y} \mapsto 3]$$

  holds ?

But how do we know it?

- Given a judgement $\langle S, s\rangle \to s'$, how do we know if it holds?
- For example, how do we know that

  $$\langle \mathtt{y} := \mathtt{y} - 1; \mathtt{skip}; \mathtt{x} := \mathtt{x} + 1, [\mathtt{x} \mapsto 3, \mathtt{y} \mapsto 4]\rangle \to [\mathtt{x} \mapsto 4, \mathtt{y} \mapsto 3]$$

  holds ?
- We define the transition relation $\to$ by a set of **rules** and **axioms**.
- These rules and axioms allow us to establish the **validity** of a judgement.

Rules and Axioms (I)

The definition of $\rightarrow$ is given by a set of **rules** and **axioms**.
A **rule** has the form:

$$[\mathit{rule\ name}] \qquad \frac{\langle S_1, s_1\rangle \rightarrow s'_1, \ \cdots \ \langle S_n, s_n\rangle \rightarrow s'_n}{\langle S, s\rangle \rightarrow s'} \quad \text{if } \mathit{condition}$$

where:

- **premises** are written above the solid line
- the **conclusion** is written below the solid line
- $S_1, \ldots, S_n$ are immediate constituents of S or constructed from immediate constituents of S
- a rule may also have a number of **conditions** or **provisos** that must be fulfilled for the rule to be applied
- if all the conditions and premises hold then the conclusion holds

Rules and Axioms (II)

The definition of $\rightarrow$ is given by a set of **rules** and **axioms**.
An **axiom** is a rule with no premises (may have conditions):

$$[axiom\ name] \qquad \frac{}{\langle S, s\rangle \rightarrow s'} \quad \text{if } condition$$

Natural Semantics for WHILE

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

The Assignment Statement :=

$$[\mathrm{ass}_{\mathrm{ns}}] \quad \langle x := a, s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$$

- executed by updating the value of x in s with the value of the arithmetic expression a in the state s

Schema vs Instance

$$[\mathrm{ass}_{\mathrm{ns}}] \quad \langle x := a, s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$$

- is a **schema**: x, s, and a are meta-variables
- we get an **instance** of the axiom when we replace the meta-variables by actual values, e.g.:

  $$\langle \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0[\mathtt{x} \mapsto 1]$$

  assuming that $s_0\ \mathtt{x} = 0$
- in general, we shall use axiom and rule to mean axiom schema and rule schema

The `skip` Statement

$$[\mathrm{skip_{ns}}] \quad \langle \mathtt{skip}, s\rangle \to s$$

- does not modify the state s

The ; Statement

$$[\text{comp}_{\text{ns}}] \quad \frac{\langle S_1, s\rangle \to s', \quad \langle S_2, s'\rangle \to s''}{\langle S_1; S_2, s\rangle \to s''}$$

- sequential composition; imposes sequential order:
  - first execute S_1 from s, obtaining s'
  - then execute S_2 from s', obtaining s''

Quiz

Assume that $s_0\ \mathtt{x} = 0$.

- is this an instance of $[\mathrm{comp_{ns}}]$?

$$\frac{\langle \mathtt{skip}, s_0\rangle \to s_0 \quad \langle \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0[x \mapsto 1]}{\langle \mathtt{skip}; \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0[x \mapsto 1]}$$

Quiz

Assume that $s_0\ \mathtt{x} = 0$.

- is this an instance of $[\mathrm{comp_{ns}}]$?

$$\frac{\langle \mathtt{skip}, s_0\rangle \to s_0 \quad \langle \mathtt{x := x + 1}, s_0\rangle \to s_0[x \mapsto 1]}{\langle \mathtt{skip; x := x + 1}, s_0\rangle \to s_0[x \mapsto 1]}$$

- is this an instance of $[\mathrm{comp_{ns}}]$?

$$\frac{\langle \mathtt{skip}, s_0\rangle \to s_0[x \mapsto 5] \quad \langle \mathtt{x := x + 1}, s_0[x \mapsto 5]\rangle \to s_0}{\langle \mathtt{skip; x := x + 1}, s_0\rangle \to s_0}$$

Quiz

Assume that $s_0\ \mathtt{x} = 0$.

- is this an instance of $[\mathrm{comp}_{\mathrm{ns}}]$?

$$\frac{\langle \mathtt{skip}, s_0\rangle \to s_0 \quad \langle \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0[x \mapsto 1]}{\langle \mathtt{skip}; \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0[x \mapsto 1]}$$

- is this an instance of $[\mathrm{comp}_{\mathrm{ns}}]$?

$$\frac{\langle \mathtt{skip}, s_0\rangle \to s_0[x \mapsto 5] \quad \langle \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0[x \mapsto 5]\rangle \to s_0}{\langle \mathtt{skip}; \mathtt{x} := \mathtt{x} + \mathtt{1}, s_0\rangle \to s_0}$$

The `if then else` Statement

We need two rules, discriminated by a condition on the guard b:

$$[\text{if}^{\text{tt}}_{\text{ns}}] \quad \frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

$$[\text{if}^{\text{ff}}_{\text{ns}}] \quad \frac{\langle S_2, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{ff}$$

- recall that a rule can only be applied if the condition is true

Quiz

We drop `then`, parenthesize b and get the rule:

$$[\mathrm{if}_{\mathrm{ns}}^{\mathrm{tt}}] \quad \frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if}\ (b)\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

is this rule valid for C, C++, or Java?

Quiz

We drop `then`, parenthesize b and get the rule:

$$[\mathrm{if}_{\mathrm{ns}}^{\mathrm{tt}}] \quad \frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if}\ (b)\ S_1\ \texttt{else}\ S_2, s\rangle \to s'} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

is this rule valid for C, C++, or Java?

Exercise

Speculative execution is an optimization technique where a computer system performs some task that may not be needed. Specify the natural semantics of the *eager execution* of the `if then else` statement.

The `while` Statement

We need an axiom:

$$[\text{while}^{\text{ff}}_{\text{ns}}] \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{ff}$$

and a rule:

$$[\text{while}^{\text{tt}}_{\text{ns}}] \quad \frac{\langle S, s\rangle \rightarrow s', \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s'\rangle \rightarrow s''}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s''} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

- the axiom formalizes termination: if b is false the loop terminates and leaves the state unchanged
- the rule formalizes looping: if b is true we execute the body and continue from a modified state

The Natural Semantics of $\texttt{WHILE}$ is not Compositional

- The culprit is the rule:

$$[\text{while}^{\text{tt}}_{\text{ns}}] \quad \frac{\langle S, s\rangle \to s', \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s'\rangle \to s''}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s''} \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

  because the semantics of `while` is defined in terms of the very same construct; not a constituent of the construct
- This means we cannot apply induction on the structure of the statements

Derivation Trees

- to validate a judgement $\langle S, s\rangle \to s'$ we build a **derivation tree**
- the **root** is the proper judgement $\langle S, s\rangle \to s'$
- the **leaves** are instances of axioms
- the **internal nodes** are conclusions of instances of rules, with the corresponding premises as children
- the **conditions** of all the instantiated axioms and rules must hold
- a derivation tree is **simple** if it is an instance of an axiom; otherwise it is **composite**

How to Build a Derivation Tree

- Given a statement S and an initial state s we proceed from the root **upwards**
- Find an axiom or rule whose conclusion matches $\langle S, s\rangle$:
  1. If it is an axiom and the condition holds, determine the final state s'. We are done.
  2. If it is a rule, recursively build derivation trees for the premises. Make sure that all the conditions hold and determine the final state.
- Note that, in general, the algorithm is **not deterministic**
- For `WHILE`, there will be at most one derivation tree

An Example of a Derivation Tree

For the statement `(z:= x; x:= y); y:= z` we get the derivation tree:

$$\dfrac{\dfrac{\langle \texttt{z:=x},\ s_0\rangle \to s_1 \qquad \langle \texttt{x:=y},\ s_1\rangle \to s_2}{\langle \texttt{z:=x; x:=y},\ s_0\rangle \to s_2} \qquad \langle \texttt{y:=z},\ s_2\rangle \to s_3}{\langle \texttt{z:=x; x:=y; y:=z},\ s_0\rangle \to s_3}$$

where $s_0\ \texttt{x} = 5$, $s_0\ \texttt{y} = 7$, $s_0\ \_ = 0$, and:

$$\begin{aligned} s_1 &= s_0[\texttt{z} \mapsto 5] \\ s_2 &= s_1[\texttt{x} \mapsto 7] \\ s_3 &= s_2[\texttt{y} \mapsto 5] \end{aligned}$$

Exercises

Exercise. Build the derivation tree for the statement:

```
y:= 1; while !(x = 1) do (y:= y*x; x:= x-1)
```

with an initial state s_0 such that $s\ \mathrm{x} = 3$.
Exercise 2.3 Build the derivation tree for the statement:

```
z:= 0; while y<=x do (z:= z+1; x:= x-y)
```

with an initial state $s_0\ \mathrm{x} = 17$ and $s_0\ \mathrm{y} = 5$.

Termination and Looping

- The execution of a statement S on a given state s
  - **terminates** if and only if there is a state s' such that $\langle S, s \rangle \to s'$, and
  - **loops** if and only if there is *no* state s' such that $\langle S, s \rangle \to s'$
- Therefore, note that no run-time errors are possible
- The execution of a statement S
  - **always terminates** if it terminates for all choices of s
  - **always loops** if it loops for all choices of s

Exercises

Exercise 2.4 Consider the following statements:

- `while !(x=1)do (y:= y*x; x:= x-1)`
- `while 1 <= x do (y:= y*x; x:= x-1)`
- `while true do skip`

For each statement determine whether or not it always terminates and whether or not it always loops. Use the axioms and rules of the natural semantics to justify your answers.

Outline

Semantic Equivalence for Natural Semantics

Two statements S_1 and S_2 are *semantically equivalent* if for all states s and s':

$$\langle S_1, s \rangle \to s' \text{ if and only if } \langle S_2, s \rangle \to s'$$

Proof Strategy: Proving Semantic Equivalence

Proving A if and only if B amounts to proving $A \Rightarrow B \wedge B \Rightarrow A$.

To prove $\langle S_1, s\rangle \to s' \Rightarrow \langle S_2, s\rangle \to s'$:

1. Assume $\langle S_1, s\rangle \to s'$
   - Apply **rule inversion** to build (*stepwise*, *bottom-up*) a derivation tree $\mathcal{D}_{S_1}$ for $\langle S_1, s\rangle \to s'$
   - The derivation tree $\mathcal{D}_{S_1}$ yields some judgements $J_1, J_2, \ldots$ known to be valid (given the assumption)
2. Prove $\langle S_2, s\rangle \to s'$
   - Using $J_1, J_2, \ldots$, apply axiom and rules to build (*stepwise*, *top-down*) a derivation tree $\mathcal{D}_{S_2}$ for $\langle S_2, s\rangle \to s'$

To prove $\langle S_2, s\rangle \to s' \Rightarrow \langle S_1, s\rangle \to s'$:

- Follow steps 1 and 2 above in reverse direction (sometimes analogous)

Your proofs **must adhere** to this strategy.

Loop Unfolding for `while`

Loop unfolding or unrolling is a loop transformation technique that attempts to optimize a program's execution speed.

Lemma 2.5 The statement

```
while b do S
```

is semantically equivalent to

```
if b then (S; while b do S) else skip
```

Proof: By construction of valid derivation trees. You must prove both directions of the equivalence.

Exercises (I)

Exercise 2.6 Prove that the two statements $S_1; (S_2; S_3)$ and $(S_1; S_2); S_3$ are semantically equivalent. Construct a statement showing that $S_1; S_2$ is not, in general, semantically equivalent to $S_2; S_1$.

Exercise 2.7 Extend the `WHILE` language with the statement

```
repeat S until b
```

and define the relation $\rightarrow$ for it. You are not allowed to rely on the `while` construct. Prove that `repeat S until b` and

```
S; if b then skip else (repeat S until b)
```

are semantically equivalent.

Exercises (II)

Exercise 2.8 Extend the `WHILE` language with the statement

```
for x := a1 to a2 do S
```

and define the relation $\rightarrow$ for it. You are not allowed to rely on the `while` construct. Evaluate the statement:

```
y:= 1; for z:= 1 to x do (y:= y*x; x:= x-1)
```

from a state where `x` has the value 5. *Hint*: Assume that you have an inverse to $\mathcal{N}$ to compute the numeral from a given number.

Deterministic Natural Semantics

A Natural Semantics is **deterministic** if for all choices of S, s, s', and s'' we have that

$$\langle S,\ s\rangle \to s' \quad \text{and} \quad \langle S,\ s\rangle \to s'' \quad \text{imply} \quad s' = s''$$

This means that for every statement S and initial state s we can uniquely determine the final state s' if (and only if) the execution of S *terminates*.

WHILE is Deterministic

Theorem 2.9 The Natural Semantics of WHILE is deterministic.
Proof: We assume that $\langle S,\ s\rangle \to s'$ and shall prove that

$$\text{if} \quad \langle S,\ s\rangle \to s'' \quad \text{then} \quad s' = s''.$$

We proceed by induction on the **shape** of the derivation tree for $\langle S,\ s\rangle \to s'$.

Induction on the Shape of the Derivation (Rule Induction)

To prove a universal property P on derivations trees:

1. Prove that the property P holds for all the *simple* derivation trees by showing that it holds for the **axioms** of the transition system.
2. Prove that the property P holds for all *composite* derivation trees: for each **rule** assume that the property holds for its premises (induction hypothesis) and prove that it also holds for the conclusion of the rule, provided that the provisos of the rule are satisfied.

Exercise

Exercise 2.10 Prove that

```
repeat S until b
```

as defined in exercise 2.7 is semantically equivalent to

```
S ; while !b do S
```

Argue that this means that the extended semantics (i.e. the natural semantics extended to include `repeat until`) is deterministic.

The Semantic Function $\mathcal{S}_{\mathbf{ns}}$

The *meaning* of statements is given by the *partial* function:

$$\mathcal{S}_{\text{ns}} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$$

This means that for every statement S we have a partial function:

$$\mathcal{S}_{\text{ns}}[\![S]\!] \in \mathbf{State} \hookrightarrow \mathbf{State}$$

defined as:

$$\mathcal{S}_{\text{ns}}[\![S]\!]s = \begin{cases} s' & \text{if } \langle S,\ s\rangle \to s' \\ \mathbf{undef} & \text{otherwise} \end{cases}$$

Quiz

We said that $\mathcal{S}_{\mathrm{ns}}$ is a partial function:

- Why is it a function?
- Why is it partial?

Quiz

We said that $\mathcal{S}_{\mathrm{ns}}$ is a partial function:

- Why is it a function?
- Why is it partial?

Exercises (I)

Exercise 2.11 The semantics of arithmetic expressions can be given by a natural semantics specification using the following two configurations:

- $\langle a,\ s\rangle$ denoting that a is to be evaluated in state s
- z denoting the final value ($z \in \mathbf{Z}$)

The transition relation $\rightarrow_{\mathbf{Aexp}}$ has the form:

$$\langle a,\ s\rangle \rightarrow_{\mathbf{Aexp}} z$$

The inference rule for addition is:

$$\frac{\langle a_1,\ s\rangle \rightarrow_{\mathbf{Aexp}} z_1, \quad \langle a_2,\ s\rangle \rightarrow_{\mathbf{Aexp}} z_2}{\langle a_1 + a_2,\ s\rangle \rightarrow_{\mathbf{Aexp}} z} \quad \text{where } z = z_1 + z_2$$

Complete the transition system of the natural semantics and use structural induction to prove that this definition is equivalent to the semantic function $\mathcal{A}$.

Exercises (II)

Exercise 2.12 We can specify the semantics for Boolean expressions using natural semantics. The transitions will have the form:

$$\langle b,\ s\rangle \rightarrow_{\mathbf{Bexp}} t$$

where $t \in \mathbf{T}$. Specify the transition system and prove that the meaning of b defined in this way is the same as that defined by $\mathcal{B}$.
Exercise 2.13 Determine whether or not semantic equivalence of S_1 and S_2 amounts to $\mathcal{S}_{\text{ns}}[\![S_1]\!] = \mathcal{S}_{\text{ns}}[\![S_2]\!]$