A Rule Format for Unit Elements

Luca Aceto
ICE-TCS, School of Computer Science, Reykjavik University

October 30, 2009

This is joint work with Anna Ingolfsdottir (RU), MohammadReza Mousavi (TU Eindhoven) and Michel A. Reniers (TU Eindhoven).

It is not possible to have a true understanding of a programming language without a mental model of its semantics, i.e., "how the language works". (Matthew Hennessy)

The General Setting

Fact of (Computer Science) Life

In Computer Science, we use formal languages to communicate with machines (programming languages) and describe expected properties of computations (specification languages).

Like natural languages, the languages we use have

1. a syntax and
2. a semantics.

Question

How are those described in CS?

The General Setting

Fact of (Computer Science) Life

In Computer Science, we use formal languages to communicate with machines (programming languages) and describe expected properties of computations (specification languages).

Like natural languages, the languages we use have

1. a syntax and
2. a semantics.

Question

How are those described in CS?

Description of Programming and Specification Languages I

Syntax

Formally specified using, e.g., BNF notation. Example?

CCS	nil $\mathbf{0}$	prefixing at
	choice $t + u$	parallel $t \| u$

where a is an action drawn from a non-empty, finite set A.

Benefits: Too many to mention! For instance, compiler technology was revolutionized and went from art to science, in the sense of Knuth.

State of Play (Syntax)

The syntax of every language under the sky is formally specified. Uncontroversial!

Semantics: Logic at Work

Operational Semantics: What the program does

Meaning of a program $\approx$ Execution on an idealized machine. How is this specified?

Plotkin's answer: Use logic! Define the semantics by using inference rules.

Operational Semantics for CCS

Given by transitions between terms of the form $t \xrightarrow{a} u$. These associate a loop-free finite automaton with each term. How?

$$\frac{}{ax \xrightarrow{a} x} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad \frac{x \xrightarrow{a} x', \; y \xrightarrow{\bar{a}} y'}{x \| y \xrightarrow{\tau} x' \| y'}$$

Semantics: Logic at Work

Operational Semantics: What the program does

Meaning of a program $\approx$ Execution on an idealized machine. How is this specified?

Plotkin's answer: Use logic! Define the semantics by using inference rules.

Operational Semantics for CCS

Given by transitions between terms of the form $t \xrightarrow{a} u$. These associate a loop-free finite automaton with each term. How?

$$\frac{}{ax \xrightarrow{a} x} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{\bar{a}} y'}{x \| y \xrightarrow{\tau} x' \| y'}$$

The Role of Equalities Between Programs

Tenet: Designers of languages often have expected algebraic properties of language constructs in mind when defining a language. Examples?

$$\begin{array}{rcl@{\qquad}rcl} x;\mathsf{skip} & = & x & x+\mathbf{0} & = & x \\ x+y & = & y+x & (x+y)+z & = & x+(y+z) \\ \mathsf{deadlock};x & = & \mathsf{deadlock} & x+x & = & x \end{array}$$

How can we ensure the validity of such laws of programming?

1. A posteriori verification: Give the semantics and then use it to prove that the laws are valid in the semantic model.
2. "Intelligent design": Give syntactic templates for the inference rules used in defining the operational semantics for certain operators that guarantee the validity of the laws by design!

The Role of Equalities Between Programs

Tenet: Designers of languages often have expected algebraic properties of language constructs in mind when defining a language. Examples?

$$\begin{array}{ccccccc} x;\mathrm{skip} & = & x & & x+\mathbf{0} & = & x \\ x+y & = & y+x & & (x+y)+z & = & x+(y+z) \\ \mathrm{deadlock};x & = & \mathrm{deadlock} & & x+x & = & x \end{array}$$

How can we ensure the validity of such laws of programming?

1. A posteriori verification: Give the semantics and then use it to prove that the laws are valid in the semantic model.
2. "Intelligent design": Give syntactic templates for the inference rules used in defining the operational semantics for certain operators that guarantee the validity of the laws by design!

Purposes of “Intelligent Design” and Challenges

1. Rule formats pave the way for a tool-set that can mechanically prove algebraic properties without involving user interaction.
2. Rule formats may serve as a guideline for language designers who want to ensure, a priori, that the constructs under design enjoy certain basic algebraic properties.

Trade-off: Generality vs. ease of application. Logic is an experimental science!

This Talk: A Rule Format for Unit Elements

A rule format guaranteeing that certain constants are left- or right-unit elements for a set of binary operators.

Technical Question: How can we guarantee the validity of equations like

$$f(c,x) = x \quad \text{and } f(x,c) = x?$$

Purposes of "Intelligent Design" and Challenges

1. Rule formats pave the way for a tool-set that can mechanically prove algebraic properties without involving user interaction.
2. Rule formats may serve as a guideline for language designers who want to ensure, a priori, that the constructs under design enjoy certain basic algebraic properties.

Trade-off: Generality vs. ease of application. Logic is an experimental science!

This Talk: A Rule Format for Unit Elements

A rule format guaranteeing that certain constants are left- or right-unit elements for a set of binary operators.

Technical Question: How can we guarantee the validity of equations like

$$f(c,x) = x \quad \text{and } f(x,c) = x?$$

Intuition

Assumption: For each constant c, we assume that each c-defining inference rule is an axiom of the form $c \xrightarrow{l} p$ for some label l and term p without variables.
Question: When is c a left unit for a binary operation f?

Wish List

1. Desideratum 1: $f(c, p)$ should be able to mimic the behaviour of p for each program p. argument.)
2. Desideratum 2: $f(c, p)$ can only mimic the behaviour of p.

How do we ensure those properties syntactically in a way that allows us to handle examples from the literature (and more)?

Intuition

Assumption: For each constant c, we assume that each c-defining inference rule is an axiom of the form $c \xrightarrow{l} p$ for some label l and term p without variables.

Question: When is c a left unit for a binary operation f?

Wish List

1. Desideratum 1: $f(c, p)$ should be able to mimic the behaviour of p for each program p. argument.)
2. Desideratum 2: $f(c, p)$ can only mimic the behaviour of p.

How do we ensure those properties syntactically in a way that allows us to handle examples from the literature (and more)?

Two Examples with a Message: Part I

Example 1

Consider the binary operators f_i, $i \geq 0$, with rules

$$\frac{x_1 \xrightarrow{a} y_1}{f_i(x_0, x_1) \xrightarrow{a} f_{i+1}(x_0, y_1)} \quad .$$

Any program is a left unit for each f_i. Why?

Lesson 1: We may need to consider possibly infinite sets of operators. Such sets cannot be defined inductively.

Two Examples with a Message: Part II

Example 2

Consider the following operations;

$$\frac{y \xrightarrow{a} y'}{f(x,y) \xrightarrow{a} g(y',x)} \qquad \frac{x \xrightarrow{a} x'}{g(x,y) \xrightarrow{a} f(y,x')}$$

Any program is a left unit for f and a right unit for g. Why?

Lesson 2: We may need to consider left and right units at the same time.

The Rule Format: Part I

Given a "language specification", the sets L and R of pairs of binary function symbols and constants are the largest sets satisfying the following constraints.

Constraint 1.L (Implements: $f(c,p)$ should be able to mimic the behaviour of p for each program p.) For each $(f,c) \in L$ and each action label a, there exists at least one rule of the following form:

$$\frac{\{x_0 \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_0 \stackrel{a_j}{\not\rightarrow} \mid j \in J\} \cup \{x_1 \xrightarrow{a} y_1\}}{f(x_0, x_1) \xrightarrow{a} t'},$$

where (1) the variables are all pairwise distinct, (2) the axioms for c "satisfy the premises involving x_0" and (3) "y_1 can be proved equal to a suitable instantiation of t'" using the laws $f(e,x) = x$ when $(f,e) \in L$ and $g(x,e') = x$ when $(g,e') \in R$.

The Rule Format: Part II

Constraint 2.L (Implements: $f(c, p)$ can only mimic the behaviour of p.) Each f-defining rule has the following form:

$$\frac{\Phi}{f(t_0, t_1) \xrightarrow{a} t'}$$

where, for each closed substitution σ such that $\sigma(t_0) \equiv c$,

1. either there exists some $t_1 \xrightarrow{a} t'' \in \Phi$ with $\sigma(t')$ "behaving like" $\sigma(t'')$, or
2. there exists a premise $\phi \in \Phi$ with t_0 as its source that cannot be met by c.

The constraints for R are symmetric.

The Syntactic Constraints At Work

Main Theorem

Consider a language with "operational semantics given by inference rules". Assume that L and R are the sets defined as given previously. For each $(f, c) \in L$, the equation $f(c, x) = x$ is valid "up to any reasonable notion of equivalence over programs". Symmetrically, for each $(f, c) \in R$, the equation $f(x, c) = x$ is valid "up to any reasonable notion of equivalence over programs".

The proof relies on the construction of suitable bisimulations.

Cool, but can it be (easily) applied to examples from the literature?

The Syntactic Constraints At Work

Main Theorem

Consider a language with "operational semantics given by inference rules". Assume that L and R are the sets defined as given previously. For each $(f, c) \in L$, the equation $f(c, x) = x$ is valid "up to any reasonable notion of equivalence over programs". Symmetrically, for each $(f, c) \in R$, the equation $f(x, c) = x$ is valid "up to any reasonable notion of equivalence over programs".

The proof relies on the construction of suitable bisimulations.
Cool, but can it be (easily) applied to examples from the literature?

Examples I

Nondeterministic Choice

Recall the rules for CCS $+$:

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

The sets $R = L = \{(+, \mathbf{0})\}$ meet the constraints. Therefore, our theorem yields the soundness of the well known equations:

$$\mathbf{0} + x = x = x + \mathbf{0}.$$

Examples II

Synchronous Parallel Composition (Hoare's CSP)

$$\frac{}{\mathrm{RUN}_a \xrightarrow{a} \mathrm{RUN}_a} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_a y \xrightarrow{a} x' \parallel_a y'} .$$

The sets $L = R = \{(\parallel_a, \mathrm{RUN}_a)\}$ meet the constraints. Therefore, our theorem yields the soundness of the well known equations:

$$\mathrm{RUN}_a \parallel_a x = x = x \parallel_a \mathrm{RUN}_a .$$

Examples III

Left Merge and Interleaving Parallel Composition

The following rules describe the operational semantics of the classic left merge and interleaving parallel composition operators.

$$\frac{x \xrightarrow{a} x'}{x \mathbin{\lfloor\!\lfloor} y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$$

The sets $L = \{(\parallel, \mathbf{0})\}$ and $R = \{(\parallel, \mathbf{0}), (\mathbin{\lfloor\!\lfloor}, \mathbf{0})\}$ meet the constraints. Therefore, our theorem yields the soundness of the well known equations:

$$\mathbf{0} \parallel x = x, \quad x \parallel \mathbf{0} = x \quad \text{and} \quad x \mathbin{\lfloor\!\lfloor} \mathbf{0} = x \ .$$

Note that the pair $(\mathbin{\lfloor\!\lfloor}, \mathbf{0})$ cannot be added to L while preserving the constraints! Indeed, $\mathbf{0}$ is not a left unit for the left merge operator $\mathbin{\lfloor\!\lfloor}$.

Examples III

Left Merge and Interleaving Parallel Composition

The following rules describe the operational semantics of the classic left merge and interleaving parallel composition operators.

$$\frac{x \xrightarrow{a} x'}{x \mathbin{\|\!\!\lfloor} y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$$

The sets $L = \{(\parallel, \mathbf{0})\}$ and $R = \{(\parallel, \mathbf{0}), (\mathbin{\|\!\!\lfloor}, \mathbf{0})\}$ meet the constraints. Therefore, our theorem yields the soundness of the well known equations:

$$\mathbf{0} \parallel x = x, \quad x \parallel \mathbf{0} = x \quad \text{and} \quad x \mathbin{\|\!\!\lfloor} \mathbf{0} = x \ .$$

Note that the pair $(\mathbin{\|\!\!\lfloor}, \mathbf{0})$ cannot be added to L while preserving the constraints! Indeed, $\mathbf{0}$ is not a left unit for the left merge operator $\mathbin{\|\!\!\lfloor}$.

Further Results

1. We have extended the rule format so that it applies to operational specifications with predicates (such as termination) as first class objects.
2. We have applied the format to more examples, such as CSP external choice, LOTOS-like disrupt, sequential composition and Plotkin's fair parallel composition operators.
3. In joint work with Matteo Cimini (Reykjavik University), we have applied our ideas to obtain a rule format for zero elements to guarantee the validity of equations like

$$\text{deadlock}; x = \text{deadlock}.$$

Thanks and Shameless Self-Promotion

Read the paper A Rule Format for Unit Elements by Anna Ingolfsdottir, MohammadReza Mousavi, Michel Reniers and yours truly! (To appear in the Proceedings of SOFSEM 2010, Springer-Verlag, 2010.)

Read the survey Algebraic Properties for Free! by Anna Ingolfsdottir, MohammadReza Mousavi, Michel Reniers and yours truly! Concurrency Column, Bulletin of the EATCS, volume 99, pp. 81–104, October 2009.

Thank You!
Any Questions?

Thanks and Shameless Self-Promotion

Read the paper A Rule Format for Unit Elements by Anna Ingolfsdottir, MohammadReza Mousavi, Michel Reniers and yours truly! (To appear in the Proceedings of SOFSEM 2010, Springer-Verlag, 2010.)

Read the survey Algebraic Properties for Free! by Anna Ingolfsdottir, MohammadReza Mousavi, Michel Reniers and yours truly! Concurrency Column, Bulletin of the EATCS, volume 99, pp. 81–104, October 2009.

Thank You!
Any Questions?