Towards Semantics Online

Peter Mosses

Swansea University
(Emeritus, visiting Delft University of Technology)

pdmosses.github.io

BCTCS 2020, Swansea, April 2020
Proposal: establish an online repository

- individual construct descriptions
  - syntax and semantics of abstract constructs
- complete language descriptions
  - translations of concrete languages to (combinations of) abstract constructs
Conclusion

Constructive semantics supports a radical change of description method:

- independent description of individual abstract constructs
- translation from concrete languages to abstract constructs

and encourages the creation of an online repository of semantic descriptions
Component-based semantics (CBS)

Conjecture

Component-based semantics can greatly reduce the effort of language specification
Component-based semantics (CBS)

Programming languages

- specified by *translation*

  to:

Components: ‘funcons’

- *fundamental* constructs

- *open-ended* library
Semantics Online requirements

Digital library

Curated repository

Language specifications

Reusable components (funcons)

Validation

Tool support
Towards Semantics Online implementation

**PLanComps**: Programming Language Components and Specifications

- 2011–2016: Swansea, RHUL, City, Newcastle

Main idea

A component-based framework

- to support design, specification, implementation of programming and domain-specific languages

with some novel aspects:
Towards Semantics Online implementation

**PLanCOMPS**: Programming Language Components and Specifications

- 2011–2016: Swansea, RHUL, City, Newcastle

- *component-based framework* (CBS meta-language, foundations)

- *specifications* (example languages, reusable components)

- *tool support* (IDE, parser generation, interpreter generation)

- *validation* (test suites)

- *historical semantic descriptions library*
Component-Based Semantics (CBS)
Funcon definitions in CBS

Based on MSOS

- **signatures**
  - distinguish between value and computation arguments

- **inductive rules** for small-step transitions
  - states: *terms*, including computed *values*
  - labels: collections of *entities* (environments, stores, signals, etc)

- *implicit propagation* of unmentioned entities
Funcon definitions in CBS

Example

- **signatures**

  Funcon \( \text{if-true-else}(_ : \text{booleans}, _ : \Rightarrow T, _ : \Rightarrow T) : \Rightarrow T \)

- **inductive rules** for small-step transitions and rewrites (\( \rightarrow \), \( \rightsquigarrow \))

  Rule \( \text{if-true-else}(B, X, Y) \rightarrow \text{if-true-else}(B', X, Y) \)

  Rule \( \text{if-true-else}(\text{true}, X, _) \rightsquigarrow X \)

  Rule \( \text{if-true-else}(\text{false}, _, Y) \rightsquigarrow Y \)
Language specifications in CBS

Languages are specified compositionally

- **context-free syntax**
  - BNF, regular expressions, disambiguation (relative priorities, etc)

- **translation functions** : syntax → funcons
  - a **semantic equation** for each language construct
  - the semantics of funcons determines the language semantics
Language specifications in CBS

Example

- context-free syntax

```
Syntax Exp : exp ::= (' exp ') | value | lexp | lexp '=' exp | '++' lexp
| '-' exp | exp (' exp ') | 'sizeof' (' exp ') | 'read' (' ')
| exp '+' exp | exp '-' exp | exp '*' exp | exp '/' exp | exp '%' exp
| exp '<' exp | exp '<=' exp | exp '>' exp | exp '>=' exp
| exp '==' exp | exp '!=' exp | '!' exp | exp '&&' exp | exp '||' exp
```
Language specifications in CBS

Example

- **translation functions**: syntax $\rightarrow$ funcons

  $\text{Semantics } rval[\_ : \text{exp}] : \Rightarrow \text{values}$

- **semantic equations**

  $\text{Rule } rval[\text{Exp}_1 \text{"&\&"} \text{Exp}_2] = \text{if-true-else}(rval[\text{Exp}_1], rval[\text{Exp}_2], \text{false})$
Modularity in CBS

Language specifications

- independent modules

Funcons library

- imported
Support for evolution in CBS

Funcon definitions

- *funcon definitions never change or disappear!*
- *new funcons can always be added*

Language specifications

- *co-evolve* with language design
- *not* reusable components
Tool support for CBS specifications

IDE for creating, editing, browsing

- grammars, translations, funcons

Generating prototypes

- language parser
- funcon interpreter
- translator : language → funcons
  - hence program execution
Recent references for CBS

Executable component-based semantics

Software meta-languages and CBS
Historical semantic descriptions
http://plancomps.org/semantic-descriptions-library/

Cliff B Jones

Semantic descriptions library

These are my (current, evolving) contributions to a "library of semantics". This material is being extended when time and resources allow. (The work was initiated during the PlanCompS project.)

Formal descriptions of ALGOL-60

Thanks to painstaking work by Roberta Velykien, the following scanned PDFs have an overlay which makes searching possible (even for Greek letters!)

- Peter Lauer's VDL description of ALGOL 60 (TR 25.088)
- A 'functional' semantics of ALGOL 60 (Notice that this scanned version deliberately omits the pages that contained the ALGOL report that were lined-up with the corresponding formulae)
- Peter Mosses' (Oxford) Denotational description of ALGOL 60
- A (actually the second) VDM description of ALGOL 60
- A re-LaTeXed version of the ALGOL 60 report
An organisation for Semantics Online
plancomps.github.io

**PLANCOMPS:** Programming Language Components and Specifications

- 2011–2016: Swansea, RHUL, City, Newcastle

Since 2016:

- more specifications (e.g., threads)
- more tool support (e.g., Markdown generation)
- a **website** for browsing languages and funcons
Languages-beta

This page provides access to the beta-release of five examples of language specifications in CBS, based on an initial collection of so-called ‘funccons’ (fundamental programming constructs). Funccons are reusable components of programming language specifications: in the CBS framework, a language is specified by defining a translation from its constructs to funccon terms, reusing (by reference) the definitions of the required funccons.

Before proceeding, take a look at the navigation and language definition sections of the Browsing guide.

See Funccons-beta for the beta-release of an initial collection of funccons. Those funcons used in the CBS of a particular language are listed in an index, accessed from the dropdown menu shown when browsing its specification.

The specification of each funccon has been validated by basic unit tests that exercise all its rules. Moreover, funcons that are reused in two or more of the illustrative language specifications have been exercised in different contexts.

The current level of validation of the illustrative language specifications varies considerably, and the coverage of the test suites has not yet been analysed. The colophon on the start page of each language specification includes the size of the current test suite and an indication of its expected coverage. Further tests are to be made during the beta-release review period; the test suites will subsequently be added to this website.
Conclusion

Towards Semantics Online – the story so far:

- 2006: Semantics Online proposed in BCTCS talk
- 2011: PLANCOMPS project started
- 2016: CBS framework established
- 2018: CBS-beta funcons and languages available for review on GitHub
- 2020: PLANCOMPS organisation on GitHub

To be continued – new participants are welcome! Email plancomps@gmail.com