

Modular Specification of Programming Languages — advances and challenges —

Peter D Mosses

Swansea University

Seminar, University of Leicester

15 October 2010

Background

Programming languages

- ▶ many **hundreds** of them
 - **older:** Fortran, Cobol, Ada, C++, Scheme, Haskell, ...
 - **newer:** Java, C#, Python, Ruby, OCaml, VBScript, ...
 - **domain-specific:** PHP, Javascript, ...
- ▶ **continually evolving**
 - new versions of existing languages
 - new language designs

Background

Programming language specifications

- ▶ **program syntax – formal**

- lexical symbols
- phrase structure

- ▶ **program semantics – informal**

- static (compile-time) behaviour
- dynamic (run-time) behaviour

Formal semantics

Problems (for major languages):

- ▶ reuse
- ▶ co-evolution
- ▶ tool support

Topics

- ▶ Structure of language specifications
- ▶ Semantic specification frameworks
- ▶ Challenges

Conventional specifications

Language semantics

Syntax

Expr

Decl

Cmd

...

Auxiliary entities

Environ

Stores

...

Semantics

Expressions

Declarations

Commands

...

Incremental specifications

Language semantics

Syntax

Auxiliary
entities

Semantics

Extended language semantics

Syntax

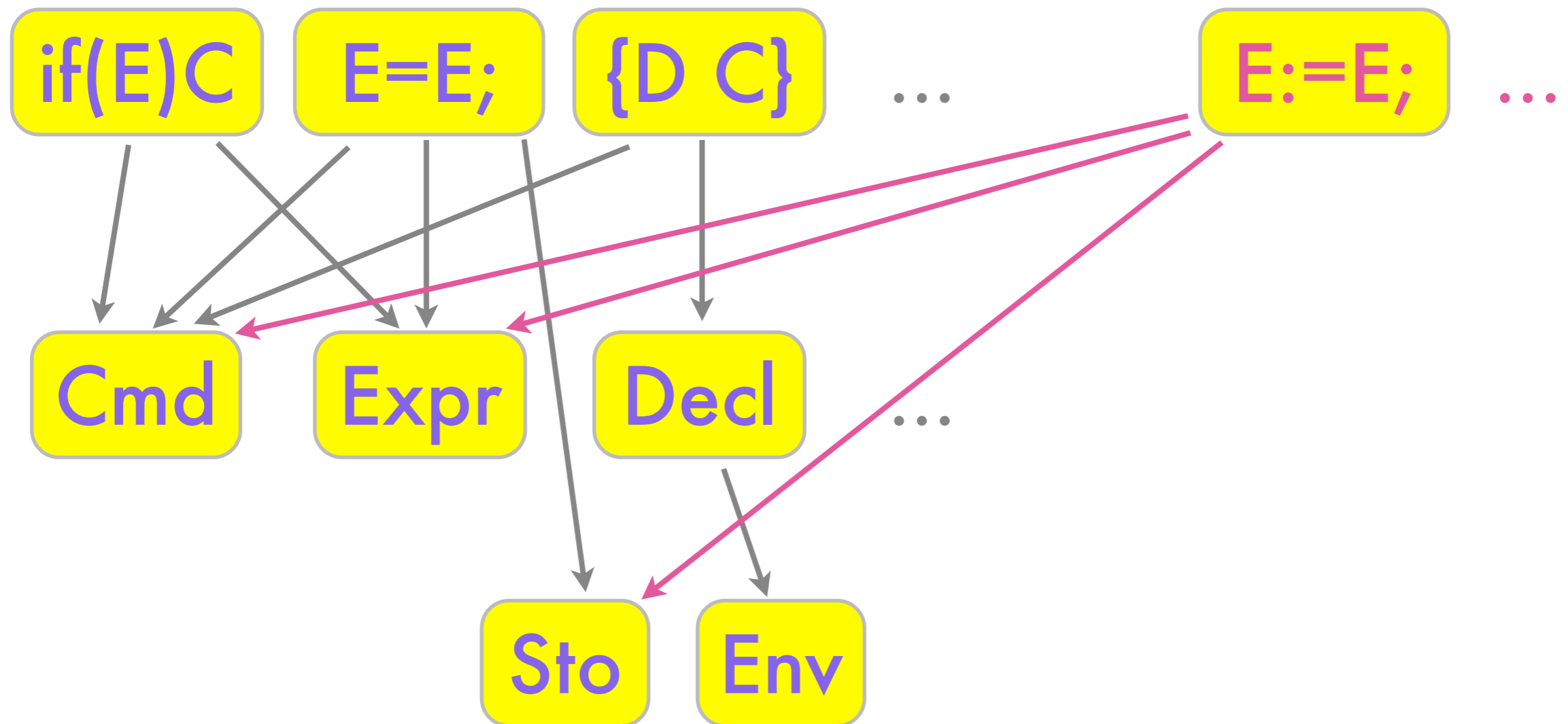
Auxiliary
entities

Semantics

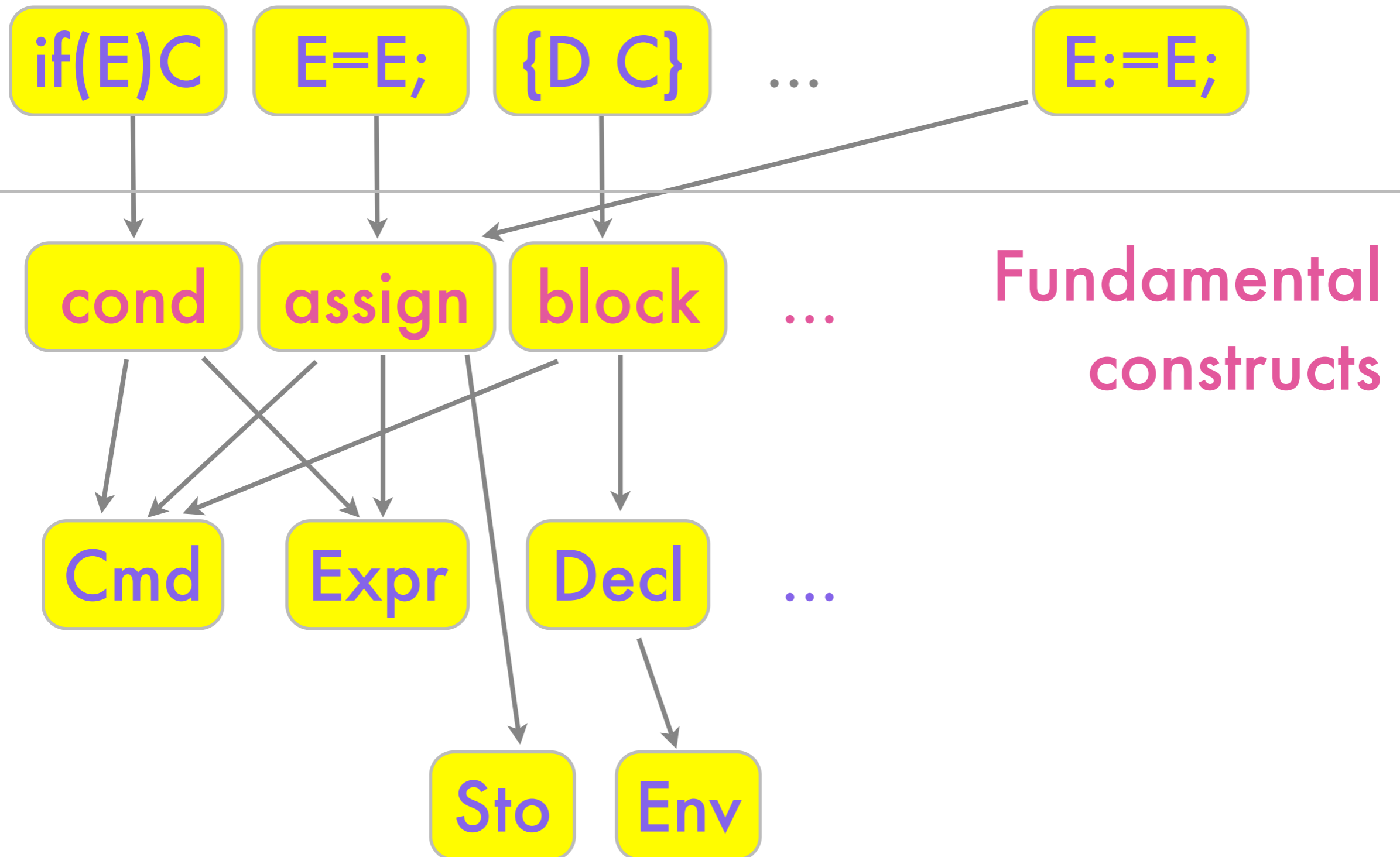
...

Component-based specifications

Language semantics



... with reusable components



Topics

- ▶ Structure of language specifications
- ▶ Semantic specification frameworks
- ▶ Challenges

Denotational semantics

Scott & Strachey (1971)

- ▶ conventional description structure
- ▶ **no** support for incremental semantics

Problem:

- ▶ **fixed interpretation of λ -notation**

$$\mathcal{C}[\varepsilon_0 := \varepsilon_1] = \lambda\rho.(\lambda\beta_0.(\lambda\beta_1.Assign(\beta_0|L, \beta_1)) * \mathcal{E}[\varepsilon_1](\rho)) * \mathcal{E}[\varepsilon_0](\rho)$$

Denotational semantics

VDM style (Bekič et al. 1974)

- ▶ conventional description structure
- ▶ **some** support for incremental semantics

Advance:

- ▶ changing interpretation of auxiliary notation

$$\begin{aligned} M[\text{mk-Assign}(lrs, rhs)](env) = \\ \quad \underline{\text{def}} \quad l: M[lhs](env); \\ \quad \underline{\text{def}} \quad v: M[rhs](env); \\ \text{STR} := \underline{c} \text{ STR} + [l \mapsto v] \end{aligned}$$

Structural operational semantics

Plotkin-style (1981), “natural” style (Kahn 1987)

- ▶ conventional description structure
- ▶ **no** support for incremental semantics

Problem:

- ▶ **explicit auxiliary information in states**

$$\frac{\rho[\rho_1] \vdash \langle C, \sigma \rangle \rightarrow \langle C', \sigma' \rangle}{\rho \vdash \langle \{ \rho_1 C \}, \sigma \rangle \rightarrow \langle \{ \rho_1 C' \}, \sigma' \rangle}$$

Abstract state machines

Gurevich et al. (mid 1980s)

- ▶ conventional description structure
- ▶ **some** support for incremental semantics

Advance:

- ▶ suppression of auxiliary entities in transition rules using imperative assignments

$\text{cond}(\blacktriangleright V_1, \beta E_2, \gamma E_3) \quad \rightarrow \quad \mathbf{if} \ V_1 \ \mathbf{then} \ pos := \beta \ \mathbf{else} \ pos := \gamma$

Action semantics

M & Watt (1985–2000)

- ▶ conventional description structure
- ▶ **good** support for incremental semantics

Advance:

- ▶ operational interpretation of auxiliary notation

execute $[[\{ D C \}]]$ = furthermore *elaborate* *D* hence *execute* *C*

Monadic semantics

Moggi (1989), Plotkin & Power (2004)

- ▶ conventional description structure
- ▶ **good** support for incremental semantics

Advance:

- ▶ flexible interpretation of auxiliary notation

$$\mathcal{C}[\{ D C \}] = \mathcal{D}[D] \gg= \lambda\rho_1. \mathbf{local}(\lambda\rho. \rho[\rho_1])(\mathcal{C}[C])$$

Modular SOS

M (1999)

- ▶ conventional description structure
- ▶ **good** support for incremental semantics

Advance:

- ▶ propagation of auxiliary entities in labels

$$\frac{C \xrightarrow{\{\rho=\rho_0[\rho_1], \dots\}} C'}{\{\rho_1 C\} \xrightarrow{\{\rho=\rho_0, \dots\}} \{\rho_1 C'\}}$$

Implicitly-Modular SOS

M & New (2008)

- ▶ good support for component-based and constructive semantics

Advance:

- ▶ implicit propagation of auxiliary entities

$$\frac{\rho[\rho_1] \vdash C \rightarrow C'}{\rho \vdash \{ \rho_1 C \} \rightarrow \{ \rho_1 C' \}}$$

Topics

- ▶ Structure of language specifications
- ▶ Semantic specification frameworks
- ▶ Challenges

The POPLmark Challenge

Pierce et al. (2005)

- ▶ Mechanized Meta-theory for the Masses

Main aim:

- ▶ machine-checked definitions and proofs about programming languages
- ▶ challenge problems involve “aspects known to be difficult to formalise” – F_{\leq} : with records

K-Challenge

Roşu (2007):

- ▶ A rewriting-based framework for computation

Main aim:

- ▶ **incremental semantics**
 - extending a simple imperative language with output, procedural abstraction, recursion, references, call/cc, nondeterminism, aspects, concurrency, synchronization, quote

Semantics-online challenge

Main aim: an **online repository**

- ▶ **reusable components**

- for language design and specification

- ▶ **major language specifications**

- exhibiting a high degree of explicit reuse

- ▶ **tool support**

- for validation and rapid prototyping

References

Scott and Strachey (1971). Towards a mathematical semantics for computer languages. In *Proc. Symp. on Computers and Automata*, Microwave Research Inst. Symposia 21:19–46. Polytechnic Institute of Brooklyn.

Bekič et al. (1974). On the formal definition of a PL/I subset (selected parts). In *Programming Languages and Their Definition - Hans Bekič (1936–1982)*, LNCS 177:107–155, 1984. Full version published as Tech. Rep. 25.139, IBM Lab. Vienna, 1974. <http://homepages.cs.ncl.ac.uk/cliff.jones/ftp-stuff/>

Plotkin (1981, 2004). A structural approach to operational semantics. *J. Log. Alg. Program.*, 60-61:17–139, 2004.

Kahn (1987). Natural semantics. In *STACS'87*, LNCS 247:22–39.

Gurevich (1987). Algebraic operational semantics. In *FSTTCS'87*, LNCS 287:1-2.

Felleisen et al. (1987). A syntactic theory of sequential control. *Theoret. Comput. Sci.* 52:205-237.

Mosses and Watt (1987). The Use of Action Semantics. In *Formal Description of Programming Concepts III*, 135–166. North-Holland.

Moggi (1989). An abstract view of programming languages. Tech. Report ECS-LFCS-90-113, Computer Science Dept., University of Edinburgh.

Plotkin and Power (2004). Computational effects and operations: An overview. In *Domains VI*, ENTCS 73:149–163.

References, ctd.

Doh and Mosses (2003). Composing programming languages by combining action-semantics modules. *Sci. Comput. Program.*, 47(1):3–36

Mosses (2004). Modular structural operational semantics. *J. Log. Algebr. Program.*, 60-61:195–228. DOI: [10.1016/j.jlap.2004.03.008](https://doi.org/10.1016/j.jlap.2004.03.008)

Iversen and Mosses (2005). Constructive action semantics for Core ML. *Software, IEE Proceedings*, 152:79–98.

Mosses (2005). A constructive approach to language definition. *J. UCS* 11(7):1117-1134.

Pierce et al. (2005). The PoplMark Challenge. http://en.wikipedia.org/wiki/POPLmark_challenge

Roşu and Serbanuta (2010). An Overview of the K Semantic Framework. *J. Log. Algebr. Program.*, 79(6):397-434.

Mosses (2008). Component-based description of programming languages. In *Visions of Computer Science*, pp. 275-286. BCS. URL: <http://www.bcs.org/server.php?show=ConWebDoc.22912>

Mosses and New (2009). Implicit propagation in structural operational semantics. In *Proc. SOS 2008, ENTCS 229(4):49-66*. DOI: [10.1016/j.entcs.2009.07.073](https://doi.org/10.1016/j.entcs.2009.07.073)

Johnstone, Mosses, and Scott (2010). An agile approach to language modelling and development. *Innovations in Systems and Software Engineering* 6(1-2):145-153. DOI: [10.1007/s11334-009-0111-6](https://doi.org/10.1007/s11334-009-0111-6)