Validity in a Modal Procedural Semantics

Giuseppe Primiero

FWO - Flemish Research Foundation
Centre for Logic and Philosophy of Science, Ghent University
IEG - Oxford University

Giuseppe.Primiero@Ugent.be
http://www.philosophy.ugent.be/giuseppeprimiero/

ECAP, Milan, Italy
6th September, 2011

Outline

Correctness in BHK ([Primiero, 2013])

- In the anti-realistic format of BHK-style semantics, logical validity is reformulated as correctness;
- Under the proofs-as-programs identity of typed systems, two readings apply:
  - semantic: "can a logical system satisfy *correctly* the typing relation for which it has been formulated?"
  - syntactic: "is it possible to formulate *correct* typing relations, so that a given validity relation is satsfied?"
- For the latter reading, the correctness relation is translated into type inhabitation for the given system;
- For dependent types, it requires formulation and access of the resources needed for a given construction.

Modalities for localized computations ([Primiero, 2011])

- Procedural Semantics with Modalities for Contextual (localized) Computing;
- designed from a multi-modal type system (BHK semantics; Proofs-as-Programs Isomorphism);
- localization of processes to represent distributed computing;
- rules for connectives intepret composition of processes;
- the behavior of programs is given by inference rules to express transition relations among states of the corresponding (abstract) machine;
- modal rules interpret interaction of code at locations (mobility).

Language

Definition (Syntax of the Programming Language)

The syntax is defined by the following alphabet:

Types $:= \alpha \mid \alpha \times \beta \mid \alpha \sqcup \beta \mid \alpha \rightarrow \beta \mid \alpha \supset \beta$

Terms $\mathcal{T} := x_i \mid a_i$

Functions $:= exec(\alpha) \mid run_i(\alpha) \mid run_{i \cup j}(\alpha \cdot \beta) \mid run_{i \cap j}(\alpha \cdot \beta) \mid synchro_j(\beta(exec(\alpha)))$

Contexts $\mathcal{C} := \Delta_i \mid \Gamma_i \mid \circ_{i,j}\Gamma$

Remote Operations $:= GLOB(\Box_{i \cup j}\Gamma, \alpha) \mid BROAD(\Diamond_{i \cap j}\Gamma, \alpha)$

Portable Code $:= RET(\Gamma_{i \cup j}, \alpha) \mid SEND(\Gamma_{i \cap j}, \alpha)$

Polymorphism

Two kinds of syntactic/semantic entities:

- the kind of specifications valid by globally terminating terms a_i;
- the kind of specifications valid by locally terminating terms x_i;

Computational Rules

Definition (Typing Rules)

$$\frac{}{\Delta_i, a_i : \alpha \vdash exec(\alpha)}\ \text{global} \qquad \frac{}{\Gamma_i, x_i : \alpha; \Delta_i \vdash run_i(\alpha)}\ \text{local}$$

$$\frac{a_i : \alpha \quad b_j : \beta}{run_{i \cup j}(\alpha \times \beta)}\ I\times \qquad \frac{a_i : \alpha}{run_i(\alpha \sqcup \beta)}\ I\sqcup$$

$$\frac{a_i : \alpha \quad exec(\alpha) \vdash b_j : \beta}{run_{i \cup j}(\alpha \to \beta)}\ I\to \qquad \frac{x_i : \alpha \quad run_i(\alpha) \vdash b_j : \beta}{run_{i \cap j}(\alpha \supset \beta)}\ I\supset$$

$$\frac{run_{i \cap j}(\alpha \supset \beta) \quad a_i : \alpha}{synchro_j(b(exec(\alpha)))}\ synchro$$

Modal Rules

Definition

$$\frac{\Gamma_i, x_j : \alpha \vdash run_j(\alpha) \quad \Box_i \Gamma, x_j(a_j) : \alpha \vdash exec(\alpha)}{GLOB(\Box_{i \cup j} \Gamma, \alpha)} \; RPC1$$

$$\frac{\Gamma_i, x_j : \alpha \vdash run_j(\alpha) \quad \Diamond_i \Gamma \vdash run_j(\alpha)}{BROAD(\Diamond_{i \cap j} \Gamma, \alpha)} \; RPC2$$

$$\frac{\Box_i \Gamma, a_j : \alpha \vdash exec(\alpha) \quad GLOB(\Box_{i \cup j} \Gamma, \alpha)}{RET(\Gamma_{i \cup j}, \alpha)} \; PORT1$$

$$\frac{\Box_i \Gamma, x_j : \alpha \vdash run_{i \cap j}(\alpha) \quad BROAD(\Diamond_{i \cap j} \Gamma, \alpha)}{SEND(\Gamma_{i \cap j}, \alpha)} \; PORT2$$

Operational Semantics

Definition (State Machine)

A state machine $S \in \mathcal{S}$

$$S := (\mathcal{C}, t.i : \alpha) \mid \mathcal{C} \in \mathit{Context};\ t \in \mathcal{T}; i \in \mathcal{I}; \alpha \in \mathit{Types}$$

is an occurrence of an indexed typed term in context.

Operational Semantics

Definition (Operational Model)

An indexed transition system (also called Network)

$$\textit{Networks}\ \mathcal{N} := (\mathcal{S}, \mapsto, \mathcal{I})$$

is a triple where $\mathcal{S}$ is a set of states, $\mathcal{I}$ is a set of indices and $\mapsto (\mathcal{S} \times \mathcal{I} \times \mathcal{S})$ is a ternary relation of indexed transitions. If $S, S' \in \mathcal{S}$ and $i, j \in \mathcal{I}$, then $\mapsto (S, i, j, S')$ is written as $S_i \mapsto S'_j$. This means that there is a transition $\mapsto$ from state S valid at index i to state S' valid at index j defined according to the state typing rules.

Evaluation

Rewriting rules for states transition:

	$S \mapsto S'$
run	$(\Gamma_i, x_i : \alpha) \mapsto (\Diamond_i \Gamma, run_i(\alpha))$
exec	$(\Gamma_i, a_i : \alpha) \mapsto (\Box_i \Gamma, exec(\alpha))$
$\rightarrow$	$(\Gamma_i, exec(\alpha) \vdash b_j) \mapsto (\Box_i \Gamma, run_{i \cup j}(\alpha \rightarrow \beta))$
$\supset$	$(\Gamma_i, run_i(\alpha) \vdash b_j) \mapsto (\Box_i \Gamma, synchro(b_j(exec(\alpha))))$
$\times$	$(\Gamma_i, exec(\alpha), exec(\beta)) \mapsto (\Box_i \Gamma, run_{i \cup j}(\alpha \times \beta))$
$\sqcup$	$(\Gamma_i, exec(\alpha)) \mapsto (\Box_i \Gamma, run_i(\alpha \sqcup \beta))$
$\Box 1$	$(\Gamma_i, exec(\alpha)) \mapsto (GLOB(\Box_{i \cup j} \Gamma, \alpha))$
$\Box 2$	$(\Box_i \Gamma, \alpha_{i \cup j}) \mapsto (RET(\Gamma_{i \cup j}, \alpha))$
$\Diamond 1$	$(\Gamma_i, run_i(\alpha)) \mapsto (BROAD(\Diamond_{i \cap j} \Gamma, \alpha))$
$\Diamond 2$	$(\Diamond_i \Gamma, \alpha_{i \cap j}) \mapsto (SEND(\Gamma_{i \cap j}, \alpha))$

Semantic Validity

Definition (Semantic Expressions)

- Evaluation defines strong typing (normalisation) by reduction to expressions $(\Box_i\Gamma, \mathit{exec}(\alpha))$ and $\mathit{GLOB}(\Box_i\Gamma, \alpha)$.
- Expressions $(\Gamma_i, \mathit{run}_i(\alpha))$ and $\mathit{BROAD}(\Diamond_i\Gamma, \alpha)$ are admissible procedural steps but may fail to produce a safe value (when called upon at wrong addresses).
- This makes (only) the following expressions valid (safely evaluated):

$$\frac{}{a_i : \alpha \ \mathit{value}} \qquad \frac{}{\Box_i\Gamma, \alpha \ \mathit{value}}$$

Failure at levels

Definition

Logical consequence is expressed as correct program execution by identifying relevant levels of information (IL) at which failure can occur:

IL1 correctness of program execution;

IL2 correctness of subcalls recursion;

IL3 correctness of data dependency;

IL4 correctness of data retrieval.

Internal Correctness

Definition

[$IL1 - 2$] identify the internal source of failure:

Internal information failure: "at which step of program execution (routines, calls for sub-routines) does the termination process fail?".

External Correctness

Definition

[$IL3 - 4$] identify the external source of failure:

> *External information failure: "which data relevant for the computational process have not been retrieved or miss appropriate dependency, so that the termination process fail?".*

Contextual Computing as framing Interaction

Interaction can be formulated as a property of contextual computational processes such that:

- it admits logical distinctions among data (data are not all the same);
- it formulates metadata (the when/how/where of data is relevant);
    - by formulating local conditions on network (processes occur as events);
    - by introducing originating locations (processes are user originated events);
- it restricts well-formedness and termination (not every process terminates or reduces).

Contextual Computing as framing Interaction

Definition (Interacting processes)

We say that a process P interacts with a process P' – denoted $Int(P, P')$ – iff at its execution, P is capable of controlling

1. access to location(s) of P';
2. commands (reading/writing/execution/broadcasting) of P';
3. validity of P' (global/local w.r.t. its locations)

and the validity of $Int(P, P')$ is determined at the union or at the intersection on their locations.

Contextual Computing as framing Interaction

This definition crucially reduces the notion of interaction to one of control over processes:

Definition (Interaction and Control)

We say that a language L expresses process interaction iff

1. L allows to represent a function $Int(P, P')$ for interaction among processes P, P';
2. $Int(P, P')$ for L allows treatment of program code accessibility; information priority and source security.

Conclusions

- A Computational Interpretation for a Multimodal Type-Theory with indexed and ordered Contexts for Mobile Processes;
- The corresponding notion of validity is treated in terms of localized correctness;
- This allows to treat notions of failure (taxonomy) and to structure interactive processes (data control).

References I

Primiero, G. (2011).
A multi-modal type system and its procedural semantics for safe distributed programming.
In *Intuitionistic Modal Logic and Applications Workshop (IMLA11)*, Nancy.
Manuscript.

Primiero, G. (Forthcoming (2013)).
Logical validity by modal types: Information control, failure and interaction.
Logique & Analyse.