Process Simulation and Refinement

He Jifeng
Oxford University Computing Laboratory Programming Research Group

Key words: Simulation; Refinement

Abstract. In this paper we deal with the problem of (nondeterministic and parallel) process refinement. The basic notion of refinement is defined via the improved failure semantics of CSP [BHR84, BrR85, Hoa85, Ros88]. The concept of simulation of Communicating Systems introduced in [Mil80, Par81] is generalised and proved to be sound for the correctness of refinement. A Galois connection is presented to show that up-simulation and down-simulation together provide a complete proof method. The paper also suggests that simulation can be employed to derive an implementation from a specification.

1. Introduction

A process is regarded as an agent which communicates with its environment by performing actions or events drawn from an alphabet A. Each event can be thought as an atomic transaction. Since we are going to be modelling processes with nondeterministic behaviour, a function (total or partial) is not enough to describe the behaviour of an action; instead a binary relation relating the states before the execution of transaction and the states afterwards will be used. This paper will present a relational model for processes.

A process P is said to be better than a process Q if P is more controllable and predictable than Q. In this case we say P refines Q. The idea of a refinement calculus is to provide a simple rule by which a refinement P of process Q can be derived (calculated) effectively. This paper will suggest two refinement rules, down-simulation and up-simulation, which will be useful both for the derivation and for the verification of an implementation.

Correspondence and offprint requests to: He Jifeng, Oxford University Programming Research Group, 8-11 Keble Road, Oxford OX1 3QD, United Kingdom.
It will be explored that both up-simulation and down-simulation are sufficient for proving a valid implementation. Further, they will generate Galois connection between the set of processes (with the preorder assigned by simulation) and the set of observations (with the refinement as its preorder). Consequently, we can show that up-simulation and down-simulation together provide a complete proof method.

The organisation of this paper is as follows. The remainder of this section consists of a few well-known concepts in category theory [HeS73, Sch53]. Section 2 presents the definition of processes, and contains an overview of the improved failure semantics of Communicating Sequential Processes [BHR84, BrR85]. Section 3 generalises the concept of simulation introduced in [Mil80, Par81], and addresses the validity of simulation rules as a refinement method. Completeness of simulations is discussed in Section 4, setting our work in a categorical context. Section 5 is the conclusion. Finally there is a technical appendix, which contains proofs of some results stated earlier.

A preorder \((S, \leq_s)\) can be considered to be a category whose objects are elements of \(S\) and for which there is one morphism from \(a\) to \(b\) just in the case \(a \leq_s b\). For preordered sets \((S, \leq_s)\) and \((T, \leq_T)\), a pair of functions
\[
\begin{align*}
f &: (S, \leq_s) \to (T, \leq_T) \\
g &: (T, \leq_T) \to (S, \leq_s)
\end{align*}
\]
is a Galois connection [Sch53] iff
\[
\begin{align*}
(1) & f \text{ and } g \text{ are order-preserving, and} \\
(2) & \text{for all } x \in S, x \leq_s gfx \text{ and all } y \in T, fgy \leq_T y.
\end{align*}
\]
Throughout this paper we will use the following conventions in notation. Given a set \(A\) of events, the set of finite sequences or traces over \(A\) is denoted \(A^*\). The empty sequence is \(\cdot\). Given traces \(s\) and \(t\), we write \(st\) for their concatenation. The powerset of \(A\) is denoted \(P(A)\). Given relations \(R\) and \(S\), \(R; S\) is the forward relational composition of \(R\) and \(S\).

2. Processes

A convenient method of modelling the step-by-step operational semantics of a process is by a labelled transition system. Processes evolve by executing actions. The nature of the actions depend on the nature of the processes under consideration. The evolution can be modelled by a relation \(\rightarrow \subseteq P \times Act \times P\), where \(P\) stands for the set of processes, and \(Act\) the set of actions. We can read
\[p \xrightarrow{a} q\]
as the process \(p\) may perform the action \(a\) and thereby be translated into the process \(q\).

In this paper we present a family \(\{P_a \subseteq P \times P \mid a \in Act\}\) of binary relations to specify the behaviour of actions performed by a process, where \(P_a\) is defined by
\[p P_a q \text{ iff } p \xrightarrow{a} q\]
In addition, a binary relation \(P_I\) is used to model the nondeterministic behaviour of the process at its very beginning. We choose a specific symbol \(\bot\) to represent