He, Jifeng; Hoare, Tony

CSP is a retract of CCS. (English) Zbl 1191.68439

Summary: Automata theory provides two ways of defining an automaton: either by its transition system, defining its states and events, or by its language, the set of sequences (traces) of events in which it can engage. For many classes of automaton, these forms of definition have been proved equivalent. For example, there is a well-known isomorphism between regular languages and finite deterministic automata. This paper suggests that for (demonically) the non-deterministic automata (as treated in process algebra), the appropriate link between transition systems and languages may be a retraction rather than an isomorphism.

A pair of automata, defined in the tradition of CCS by their transition systems, may be compared by a pre-ordering based on some kind of simulation or bisimulation, for example, weak, strong, or barbed. Automata defined in the tradition of CSP are naturally ordered by set inclusion of their languages (often called refinement); variations in ordering arise from different choices of basic event, including for example, refusals and divergences. In both cases, we characterise a theory by its underlying transition system and its choice of ordering. Our treatment is therefore wholly semantic, independent of the syntax and definition of operators of the calculus.

We put forward a series of retractions relating the above-mentioned versions of CSP to their corresponding CCS transition models. A retraction is an injection that is (with respect to the chosen ordering) monotonic, increasing and idempotent (up to equivalence). It maps the nodes of a transition system of its source theory to those of a system that has been saturated by additional transitions. Each retraction will be defined by a transition rule, in the style of operational semantics; the proofs use the familiar technique of co-induction, often abbreviated by encoding in the relational calculus.

The aim of this paper is to contribute to unification of theories of reactive system programming. More practical benefits may follow. For example, we justify a method to improve the efficiency of model checking based on simulation. Furthermore, we show how model checking of a transition network fits consistently with theorem-proving tools, which reason directly about specifications and designs that are expressed in terms of sets of sequences of observable events.

MSC:
68Q85 Models and methods for concurrent and distributed computing (process algebras, bisimulation, transition nets, etc.)
68Q45 Formal languages and automata

Keywords:
unifying theories of programming; process algebra; CCS; CSP; retraction

Software:
Zing

Full Text: DOI

References:
Zbl 0579.68016


[18] Phillips, I., (), 304-313


[20] Rajamani, Srinivas K.; Rehof, Jakob; Qadeer, Shaz; Xie, Yichen; Andrews, Tony, Zing: A model checker for concurrent software, (), 484-487 · Zbl 1103.68600


[22] ()


[26] van Glabbeek, R.J., The linear time — branching time spectrum II; the semantics of sequential processes with silent moves, (), 66-81, (extended abstract)

This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.