Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 250 (2009) 69–84

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

CSP-CASL-Prover: A Generic Tool for Process and Data Refinement

Liam O'Reilly[1] Markus Roggenbach[1]

Swansea University, United Kingdom

Yoshinao Isobe[1,2]

AIST, Tsukuba, Japan

Abstract

The specification language Csp-Casl allows one to model processes as well as data of distributed systems within one framework. In our paper, we describe how a combination of the existing tools Hets and Csp-Prover can solve the challenges that Csp-Casl raises on integrated theorem proving for processes and data. For building this new tool, the automated generation of theorems and their proofs in Isabelle/HOL plays a fundamental role. A case study of industrial strength demonstrates that our approach scales up to complex problems.

Keywords: Process Algebra, Algebraic Specification, Theorem Proving, Functional Programming.

1 Introduction

Distributed computer applications like flight booking systems, web services, and electronic payment systems such as the EP2 standard [2], require parallel processing of data. Consequently, these systems have concurrent aspects (e.g. deadlock-freedom) as well as data aspects (e.g. functional correctness). Often, these aspects depend on each other.

In [22], we present the language Csp-Casl, which is tailored to the specification of distributed systems. Csp-Casl integrates the process algebra Csp [7,23] with the algebraic specification language Casl [15]. Its novel aspects include the combination of denotational semantics in the process part and, in particular, loose semantics for

[1] This cooperation was supported by the EPSRC Project EP/D037212/1.

[2] This work was supported by KAKENHI 20500023.

the data types covering both concepts of partiality and sub-sorting. In [5] we apply Csp-Casl to the EP2 standard and demonstrate that Csp-Casl can deal with problems of industrial strength.

Here, we develop theorem proving support for Csp-Casl and show that our approach scales up to practically relevant systems such as the EP2 standard. Csp-Casl comes with a simple, but powerful notion of refinement. Csp-Casl refinement can be decomposed into first a refinement step on data only and then a refinement step on processes. Data refinement is well understood in the Casl context and has good tool support already. Thus, we focus here on process refinement. The basic idea is to re-use existing tools for the languages Casl and Csp, namely for Casl the tool Hets [13] and for Csp the tool Csp-Prover [8,9,10,11], both of which are based on the theorem prover Isabelle/HOL [19]. This re-use is possible thanks to the definition of the Csp-Casl semantics in a two step approach: First, the data specified in Casl is translated into an alphabet of communications, which, in the second step, is used within the processes, where the standard Csp semantics are applied.

The main issue in integrating the tools Hets and Csp-Prover into a Csp-Casl-Prover is to implement – in Isabelle/HOL – Csp-Casl's construction of an alphabet of communications out of an algebraic specification of data written in Casl. The correctness of this construction relies on the fact that a certain relation turns out to be an equivalence relation. [22] shows in terms of a manually proven meta theorem that the alphabet construction works out for a large class of Casl data specifications, which is characterised by the static semantics property 'has local top elements'. In Csp-Casl-Prover, we choose to prove the relation to be an equivalence for each Csp-Casl specification individually. This adds an additional layer of trust: complementing the algorithmic check of a static property, we provide a proof in Isabelle/HOL that the construction is valid. The alphabet construction, the formulation of the justification theorems (establishing the equivalence relation), and their proofs can all be automatically generated.

Closely related to Csp-Casl is the specification language μCRL [4]. Here, data types have loose semantics and are specified in equational logic with total functions. The underlying semantics of the process algebraic part is operational. [1] presents a μCRL-Prover based on the interactive theorem prover PVS. The chosen approach is to represent the abstract μCRL data types directly by PVS types, and to give a subset of μCRL processes an operational semantics. Thanks to μCRL's simple approach to data – neither sub-sorting nor partiality are available – there is no need for an alphabet construction – as it is also the case in Csp-Casl in the absence of sub-sorting and partiality. Concerning processes, Csp-Casl-Prover provides semantics to full Csp.

Our paper is organised as follows: Section 2 introduces the Csp-Casl semantics along with a case study from the EP2 system. Section 3 describes the existing tools which we make use of. The overall architecture of Csp-Casl-Prover is presented in Section 4. First we discuss how to build an alphabet to be used as a parameter for the process type of Csp-Prover. Then we consider how integration theorems can