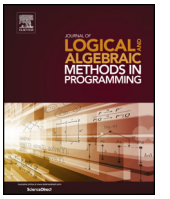
Journal of Logical and Algebraic Methods in Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



Observational and behavioural equivalences for soft concurrent constraint programming ☆

Fabio Gadducci [a,*], Francesco Santini [b,**], Luis F. Pino [c], Frank D. Valencia [d,e]

[a] Dipartimento di Informatica, Università di Pisa, Italy
[b] Dipartimento di Matematica e Informatica, Università di Perugia, Italy
[c] Dipartimento di Matematica e Informatica, Università di Cagliari, Italy
[d] CNRS and LIX, Ecole Polytechnique de Paris, France
[e] Pontificia Universidad Javeriana de Cali, Colombia

ARTICLE INFO

Article history:
Received 9 May 2016
Received in revised form 11 June 2017
Accepted 12 June 2017
Available online xxxx

ABSTRACT

We present a labelled semantics for Soft Concurrent Constraint Programming (SCCP), a meta-language where concurrent agents may synchronise on a shared store by either posting or checking the satisfaction of (soft) constraints. SCCP generalises the classical formalism by parametrising the constraint system over an order-enriched monoid, thus abstractly representing the store with an element of the monoid, and the standard unlabelled semantics just observes store updates. The novel operational rules are shown to offer a sound and complete co-inductive technique to prove the original equivalence over the unlabelled semantics. Based on this characterisation, we provide an axiomatisation for finite agents.

1. Introduction

Concurrent Constraint Programming (CCP) [1] is a language based on a shared-memory communication pattern: processes may interact by either posting or checking partial information, which is represented as constraints in a global store. CCP belongs to the larger family of process calculi, thus a syntax-driven operational semantics represents the computational steps. For example, the term **tell**(c) represents a process that posts c in the store, and the term **ask**(c) $\rightarrow$ P is the process that executes P if c can be derived from the information in the store.

The formalism is parametric with respect to the entailment relation. Under the name of *constraint system*, the information recorded on the store is structured as a partial order (in fact, a lattice) $\leq$, where $c \leq d$ means that c can be derived from d. Under a few requirements over such systems, CCP has been provided with (coincident) operational and denotational semantics. A simple example of interaction is given by the processes

$$P_1 : \mathbf{tell}(X = 30)$$

☆ The research has been partially supported by the MIUR PRIN 2010LHT4KM CINA and PRIN 2010XSEMLC "Security Horizons", by the ANR 12IS02001 PACE, and by the Aut. Reg. of Sardinia P.I.A. 2010 "Social Glue".

* Principal corresponding author.
** Secondary corresponding author.
E-mail addresses: fabio.gadducci@unipi.it (F. Gadducci), francesco.santini@dmi.unipg.it (F. Santini), luis.pino@unica.it (L.F. Pino), frank.valencia@lix.polytechnique.fr (F.D. Valencia).

http://dx.doi.org/10.1016/j.jlamp.2017.06.001
2352-2208/

$$P_2 : \mathbf{ask}(X > 19 \wedge X < 21) \rightarrow \mathbf{tell}(Y = 0) + \mathbf{ask}(X < 19 \vee X > 21) \rightarrow \mathbf{tell}(Y = 1)$$

where a constraint is just a function associating to X and Y a finite set of natural numbers, the order is induced by (the inverse of) subset inclusion, and telling a constraint to the store is based on subset intersection. Thus, the parallel composition $P_1 \parallel P_2$ represents a sensor reporting the current temperature (represented by variable X), and an air conditioning system that activates if the temperature is below 19 or above 21 degrees Celsius (since Y represents an on/off switch). In this case, $(X = 30) \geq (X < 19 \vee X > 21)$, so that P_2 may be stuck until P_1 posts its information to the store.

A key aspect of CCP is the *idempotency* of the operator for composing constraints: adding the same information twice does not change the store. For instance, $\mathbf{tell}(X = 30) \parallel \mathbf{tell}(X = 30)$ leads to a store where the temperature is still 30 degrees. On the contrary, the soft variant of the formalism (Soft CCP, or just SCCP [3]) drops idempotency: constraint systems in SCCP may distinguish the number of occurrences of a piece of information. If, for example, a preference for a constraint c is directly linked to the temperature it carries as information, then we have that $\mathbf{tell}(c) \parallel \mathbf{tell}(c)$ results in having a preference as for constraint that reports 60 degrees. Dropping idempotency requires a complete reworking of the theory. Although an operational semantics for SCCP has been devised [3], hitherto neither the denotational nor the labelled one has been reintroduced. This is unfortunate, since due to its generality suitable SCCP instances has been successfully applied as a specification formalism for negotiation of Service Level Agreements [4], or the enforcement of ACL-based access control [5].

As a language, SCCP has been used as a specification formalism for agents collaborating via a shared knowledge basis, possibly with temporal features [6,7]. Thus, on a methodological level, the development of behavioural equivalences for SCCP may result in the improvement on the analysis techniques for agents that need to reason guided by their preferences, more so if their knowledge is not complete.

In more general terms, SCCP represents, by its parametric nature, a formal meta-model where to develop different constraint-languages over different (weighted or crisp) logics, as it will clearly appear from Section 3.

The work in [1] establishes a denotational semantics for CCP and an equational theory for infinite agents. More recently, in [2] the authors prove that the axiomatisation is underlying a specific weak bisimilarity among agents, thus providing a clear operational understanding. The key ingredients are a complete lattice as the domain of the store, with least upper bound for constraint combination, and a notion of compactness such that domain equations for the parallel composition of recursive agents would be well-defined. On the contrary, the soft version [3] drops the upper bound for combination in exchange of a more general monoidal operator. Thus, the domain is potentially just a (not necessarily complete) partial order, possibly with finite meets and a residuation operator (a kind of inverse of the monoidal one) in order to account for algorithms concerning constraint propagation. Indeed, the main use of SCCP has been in the generalisation of classical constraint satisfaction problems, hence the lack of investigation about e.g. compactness and denotational semantics.

The objective of our work is the development of a general theory for the operational semantics of SCCP, via the introduction of suitable observational and behavioural equivalences. Reaching this objective is technically challenging, since most of the simplicity of CCP is based precisely on the premise that posting an information multiple times is the same as posting it only once. The first step consists in recasting the notion of compactness from crisp to soft; we then introduce a novel labelled semantics for SCCP which will allow us to give a sound and complete technique to prove the equivalence over the unlabelled semantics.

We will build our framework by supposing to have a global store, that is shared by all the agents. Such a premise is required by how we design the transition system: in fact, it is labelled with a shared set of variables (i.e., Δ) as a means to keep track of variables' name after renaming them through the hiding operator. Since variables support the constraints posted to the store, renaming is more subtle than in other algebras: the store retains a memory of former names. In the future we plan to investigate how hiding can be performed in case of stores that are local to each agent (see Section 8).

This paper details the work in [8] and extends it by reconnecting the presented framework with the classical work on soft constraint systems [3] (see Section 2.4). Section 2 opens the paper with the technical background, presenting also some novelty as $\otimes$-compact elements. Section 3 presents the semantics of a deterministic fragment of a constraint language, together with fundamental concepts, e.g., observables, confluence, observational equivalence, and, in Section 4, the notion of saturated bisimulation. Section 5 derives a labelled transition system for SCCP, soundness and completeness with respect to the unlabelled one, and weak/strong bisimilarity relations. Section 6 shows a sound and complete axiomatisation for a finite fragment of the language. Finally, Section 7 reports the main literature in the field, while Section 8 wraps up the paper with conclusions and future work.

2. The algebraic background

This section recalls the main notions we are going to need later on. First of all, we present some basic facts concerning monoids [9] enriched over complete lattices. These are used to recast the standard presentation of the soft constraints paradigm, and to generalise the classical crisp one.

2.1. Lattice-enriched monoids

Definition 1 *(Complete lattices).* A partial order (PO) is a pair $\langle A, \leq \rangle$ such that A is a set of values and $\leq \subseteq A \times A$ is a reflexive, transitive, and anti-symmetric relation. A complete lattice (CL) is a PO such that any subset of A has a least upper bound (LUB).

Please cite this article in press as: F. Gadducci et al., Observational and behavioural equivalences for soft concurrent constraint programming, J. Log. Algebraic Methods Program. (2017), http://dx.doi.org/10.1016/j.jlamp.2017.06.001