Verification and Debugging of Concurrent Constraint Programs through Abstract Interpretation with Set Constraints

Andreas Podelski
Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken
podelski@mpi-sb.mpg.de

Abstract

The existing automated verification methods apply mainly to those concurrent systems where the number of concurrent processes is statically fixed and the data dependencies of the control flow are simple. Systems that are specified and programmed in the concurrent constraint programming paradigm (cc) specifically overcome those two limitations. There, memory states are modeled by logical formulae (constraint stores); these are conjunctions which monotonically grow in one transition sequence. We give a framework for specifying correctness criteria for cc systems (with an unbounded number of processes). We give a method for verifying or debugging temporal-logic properties of cc systems in which we extract a set constraint syntactically from the program and the correctness specification. That set constraint is a high-level specification of the automaton which accepts a (lower or upper, respectively) approximation of the set of correct input states.

Our method relies on two observations that we establish in this paper: (1) Temporal properties are characterized through least and greatest fixpoints of the T_P operator from constraint logic programming (if the underlying constraint domain has the so-called saturation property). (2) The greatest model semantics, *i.e.*, $gfp(T_P)$, can be approximated by a set constraint. We formalize our approximation process in the abstract interpretation framework. For the least model semantics, its approximation by set constraints was already introduced by Heintze and Jaffar and formalized as an abstract interpretation by the Cousots.

1 Overview

We show that temporal properties of (terminating or non-terminating) executions of a constraint logic program P are characterized through the least and greatest fixpoints of the T_P operator (T_P maps a set of ground facts to the set of their immediate logical consequences under P). The characterization puts a requirement on the underlying constraint domain (the saturation property, see below).

As a consequence of the characterization, one can use existing or new methods of abstract interpretation with respect to the least or greatest model semantics of CLP programs for approximating temporal properties from the above or the below and obtain automated tools for the verification or the debugging, respectively, of cc programs. Given a concurrent constraint program P_{cc}, we first approximate its set of (generally non-terminating) executions by the one of an associated constraint logic program P.

We give a new method for approximating the greatest model semantics, *i.e.*, $gfp(T_P)$, through a set constraint $\mathcal{K}_P^{\cup}$ (its greatest solution, precisely). As usual in set-based analysis, the set constraint is derived syntactically from the program. We establish the approximation, however, as an abstract interpretation process, und use this to formalize it, prove it correct and define the degree of approximation.

For the first time, set constraints are used for the approximation of the greatest model semantics. Previous approaches have considered only the case of the least model semantics. Also, for the first time, set-based analysis works on a parametrized constraint domain. Previous approaches have mainly considered the case of the domain of finite trees. We give two conditions which define when an extension of the notion of constraints on a parametrized constraint domain D to a notion of set constraints (*i.e.*, a logic over the domain $\mathcal{P}(\mathcal{D})$ of sets of values in $\mathcal{D}$) is appropriate for the set-based analysis considered here. We define a logic of set constraints over sets of (finite, rational or infinite) trees which fulfills this requirement.

The characterization of temporal properties through $gfp(T_P)$ requires an assumption on the underlying constraint domain. Namely, if every finite subset of a set S of constraints is satisfiable, then also the set S. This property is called *saturation*.

The constraint domain of the concurrent constraint programming language Oz takes equations over infinite trees.[1] This domain owns the saturation property. This has already been shown by Palmgren (1994). We can give a new proof which in a simple way exploits the topological properties of the constraint domain (which is compact, and the solutions of a constraint form a closed set).

Now, we can instantiate our framework to the constraint domain of infinite trees. Since a set constraint can be tested for consistency and, if satisfiable, brought into a solved form representing the greatest solution (essentially as a tree automaton) in worst-case exponential time (this is also a lower bound), we obtain an automated procedure for approximating temporal properties of cc programs. If the tree automaton used for the representation is deterministic, the test on a given state with constraint store φ is linear in φ.

2 Example

We take the example of a stream program in CLP-like syntax.

$$p(x) \leftrightarrow \exists y \exists z \;\; x = [y|z] \;\wedge\; \texttt{<computation on } y \texttt{ and } z\texttt{>} \;\wedge\; p(z)$$

We may be interested in states where the procedure p is called and the constraint store is compatible with the fact that the actual argument of p lies in a particular set of values. We call the set of such states Prop. We may want to check for some given state s whether for some (for all) executions starting in s all states reached lie in Prop (or whether a state in Prop can be reached). The four properties of the state s that we may want to check correspond to membership in the set obtained by applying one of four temporal logic operators on the set Prop.

3 Set constraints for greatest model semantics

We assume a constraint system consisting of a class $\mathcal{L}$ of constraints φ (*i.e.*, a subclass of first-order formulas over a given signature Σ which is closed under conjunction and existential quantification) and a structure $\mathcal{D}$ (*i.e.*, a domain D of values together with an interpretation of the function and relation symbols occurring in Σ).

We next define the class $\mathcal{L}^{\cup}$ of *set constraints $\mathcal{K}$ with union* by extending the syntax of the constraints φ with the function symbol $\cup$ and the relation symbol $\subseteq$.

$$\mathcal{K} \;::=\; \varphi \mid x \subseteq y \mid x = y \cup z \mid \mathcal{K}_1 \wedge \mathcal{K}_2$$

[1] In sequential CLP languages (*e.g.*, Prolog-II), one enlargens the domain of finite trees with rational trees in order to account for cyclic data structures. The choice of infinite trees, as opposed to just rational ones, is due to the fact that these can be used to describe the limits of non-terminating executions.

The intended interpretation domain of set constraints $\mathcal{K}$ is the domain $\mathcal{P}(D)$ of sets of values of D. There is no generic way to define their interpretation for a parametrized constraint system. It seems natural to require that the singletons of solutions of constraints φ over D are solutions of φ (now as a set constraint) over $\mathcal{P}(D)$. Formally:

Condition 1. If $\alpha : \mathsf{Var} \to \mathcal{D}$ is a solution of the constraint φ, then the valuation $\sigma : \mathsf{Var} \to \mathcal{P}(D)$ defined by $\sigma(x) = \{\alpha(x)\}$ is a solution of the set constraint φ.

In our framework we require a second condition, namely that solutions of set constraints are closed under union. (Roughly, this means that the interpretation is "coarse enough". In the examples we have considered it is obtained by introducing approximations.)

Condition 2. If σ_1 and σ_2 are solutions of the set constraint $\mathcal{K}$ then also the valuation $\sigma : \mathsf{Var} \to \mathcal{P}(D)$ defined by $\sigma(x) = \sigma_1 \cup \sigma_2$ is a solution of $\mathcal{K}$.

We say that the constraint system consisting of $\mathcal{L}^{\cup}$ and the structure $\mathcal{D}^{Sets}$ with the domain $\mathcal{P}(D)$ of sets of values in $\mathcal{D}$ is *appropriate* if Conditions 1 and 2 hold.

A CLP program consists of clauses of the form below, where p is a predicate ($p \in \mathsf{Proc}$) defined by, say, n clauses which are all renamed apart, *i.e.*, don't share any variables (and $i = 1, \ldots, n$).

$$p(\bar{x}_p^i) \leftarrow \psi_i \wedge \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}))$$

For the logical semantics, we refer to the following formula (in which we leave the existential quantifiers implicit; the variables shared among disjuncts are only the variables in the tuple $\bar{x}_p$ in the definition of the predicate $p(\bar{x}_p)$).

$$P \equiv \bigwedge_{p \in \mathsf{Proc}} p(\bar{x}_p) \leftrightarrow \bigvee_{i=1,\ldots n} (\bar{x}_p = \bar{x}_p^i \wedge \psi_i \wedge \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}))$$

We map P to the following set constraint $\mathcal{K}_P^{\cup}$.

$$\mathcal{K}_P^{\cup} \equiv \bigwedge_{p \in \mathsf{Proc}} \bar{x}_p = \bigcup_{i=1,\ldots n} \bar{x}_p^i \wedge \bigwedge_{i=1,\ldots n} (\psi_i \wedge \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \bar{x}_{p_{ij}})$$

In the special case where the constraint domain is $\mathsf{Herbrand}$, $\mathcal{K}_P$ coincides with the set constraint obtained from a Prolog program by the type inference algorithm of Misra (1984); there, however, the inferred set constraint is used to approximate the *least* model (*i.e.*, the least fixpoint of the T_P operator).

We recall that T_P maps a set X of ground facts to the set of their immediate consequences under the programm P; *i.e.*, if $X \subseteq \mathcal{B}_{\mathcal{D}} = \{p(\bar{d}) \mid p \in \mathsf{Proc},\ \bar{d} \in \bar{\mathcal{D}}\}$, then

$$T_P(X) = \{p(\bar{d}) \mid \exists \alpha : \mathsf{Var} \to D\ \exists i \in \{1, \ldots, n\} : \begin{array}{l} \alpha(\bar{x}_p^i) = \bar{d}, \\ \mathcal{D}, \alpha \models \psi_i, \\ p_{ij}(\alpha(\bar{y}_{ij})) \in X\} \end{array}$$

where the set comprehension ranges over all predicates $p \in \mathsf{Proc}$ defined by the program P.

Theorem 1 If the system $\langle \mathcal{L}^{\cup}, \mathcal{D}^{Sets} \rangle$ of set constraints with union is an appropriate extension of the system of constraints that underlies the program P, then the greatest model of P (or, equivalently, the greatest fixpoint of T_P) is approximated by the greatest solution of K_P.

$$gfp(T_P) \subseteq \{p(\bar{d}) \mid p \in \mathsf{Proc},\ \bar{d} \in \mathsf{greatestSolution}(\mathcal{K}_P^{\cup})(\bar{x}_p)\}$$

If we set $p^{-1}(S) = \{\bar{d} \mid p(\bar{d}) \in S\}$ for a set S and a predicate p, then the inclusion above can be written as the conjunction of the inclusions below (for all $p \in \mathsf{Proc}$).

$$p^{-1}(gfp(T_P)) \subseteq \mathsf{greatestSolution}(\mathcal{K}_P^{\cup})(\bar{x}_p)$$

Proof. We will prove Theorem 1 after we state Theorem 2.

We say that a subset X of $\mathcal{B}_\mathcal{D}$ is *Cartesian* if there exist subsets $D_{p,1}, \ldots, D_{p,l} \subseteq D$ such that

$$p^{-1}(X) = D_{p,1} \times \ldots D_{p,l}$$

(for all $p \in \mathsf{Proc}$, where l is the arity of p). We represent X by a mapping $\rho : \mathsf{Var} \to \mathcal{P}(D)$ that maps each formal variable $x_{p,k}$ (the k-th formal argument of the l-ary procedure p) to $D_{p,k}$ and each other variable to the whole domain D, thus representing X by

$$X = \bigcup_{p \in \mathsf{Proc}} p(\rho(x_{p,1}) \times \ldots \times \rho(x_{p,l})).$$

We define the abstract domain $\mathcal{P}(\mathcal{B}_\mathcal{D})^\#$ as the subdomain of the concrete domain $\mathcal{P}(\mathcal{B}_\mathcal{D})$ consisting of all its Cartesian sets. The *abstraction of the T_P operator* is an operator defined on $\mathcal{P}(\mathcal{B}_\mathcal{D})^\#$ by

$$\begin{aligned} T_P^\#(X) = \{p(d_1, \ldots, d_l) \mid\ & \forall k = 1, \ldots, l \\ & \exists \sigma : \mathsf{Var} \to \mathcal{P}(D)\ \exists i \in \{1, \ldots, n\} : \quad d_k \in \sigma(x^i_{p,k}), \\ & \qquad\qquad\qquad\qquad\qquad\quad \mathcal{D}^{Sets}, \sigma \models \psi_i, \\ & \qquad\qquad\qquad\qquad\qquad\quad p_{ij}(\sigma(\bar{y}_{ij})) \subseteq X\}. \end{aligned}$$

Clearly, the image X' of a Cartesian set X under $T_P^\#$ is again Cartesian. It is represented by ρ' where

$$\begin{aligned} \rho'(x_{p,k}) = \{d \in D \mid\ & \exists \sigma : \mathsf{Var} \to \mathcal{P}(D)\ \exists i \in \{1, \ldots, n\} : \quad d \in \sigma(x^i_{p,k}), \\ & \qquad\qquad\qquad\qquad\qquad\quad \mathcal{D}^{Sets}, \sigma \models \psi_i, \\ & \qquad\qquad\qquad\qquad\qquad\quad p_{ij}(\sigma(\bar{y}_{ij})) \subseteq X\}. \end{aligned}$$

The following theorem formalizes the process of approximation in the previous theorem and motivates the definition of the mapping $P \mapsto \mathcal{K}_P^\cup$.

Theorem 2 The greatest solution of $\mathcal{K}_P$ is the greatest fixpoint of the operator $T_P^\#$.

$$\{p(\bar{d}) \mid p \in \mathsf{Proc},\ \bar{d} \in \mathsf{greatestSolution}(\mathcal{K}_P^\cup)(\bar{x}_p)\} = gfp(T_P^\#)$$

Another way to write the inclusion above is as the conjunctions of the inclusions

$$\mathsf{greatestSolution}(\mathcal{K}_P^\cup)(\bar{x}_p) = p^{-1}(gfp(T_P^\#))$$

for all $p \in \mathsf{Proc}$.

Condition 1 implies immmediately the soundness of the approximation of T_P by $T_P^\#$, *i.e.*,

$$T_P \circ \gamma \subseteq \gamma \circ T_P^\#$$

where the concretization function from $\mathcal{P}(\mathcal{B}_\mathcal{D})^\#$ to $\mathcal{P}(\mathcal{B}_\mathcal{D})$ is the identity. Thus, in the abstract interpretation framework of the Cousots, Theorem 1 is a direct consequence of Theorem 2.

Proof. [of Theorem 2]Let $X \in \mathcal{P}(\mathcal{B}_\mathcal{D})^\#$ be represented by the mapping $\rho : \mathsf{Var} \to \mathcal{P}(D)$ and its image $X' = T_P^\#(X)$ of X under the abstraction of the T_P operator by the mapping ρ'. We may represent the value of the variable $x_{p,k}$, the k-th formal argument of the procedure p under the mapping ρ' as follows.

$$\begin{aligned} \rho'(x_{p,k}) = \bigcup_{\sigma \in \mathsf{Var} \to \mathcal{P}(D)} \bigcup_{i \in \{1,\ldots,n\}} \{\sigma(x^i_{p,k}) \mid\ & \mathcal{D}^{Sets}, \sigma \models \psi_i, \\ & \textstyle\bigwedge_{j=1,\ldots,m_i} \sigma(\bar{y}_{ij})) \subseteq \rho(\bar{x}_{p_{ij}})\} \end{aligned}$$

If we assume that *all constraints ψ_i are satisfiable*, then, since we have assumed that the system $\mathcal{D}^{Sets}$ is appropriate, *i.e.*, it satisfies Condition 2, we may transform the above equality to the following one.

$$\rho'(x_{p,k}) = \mathsf{greatestSolution}(\bar{x}_p = \bigcup_{i\in\{1,\ldots,n\}} \bar{x}_p^i \wedge \bigwedge_{i=1,\ldots,n} (\psi_i \wedge \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \rho(\bar{x}_{p_{ij}})(x_{p,k})))$$

Note that the greatest solution of the conjunctions of set constraints with conjunctions of the form $x \subseteq S$ for sets $S \subseteq D$ exists since it exists for set constraints due to Condition 2. We may assume that the equality holds for all variables and write an equality between two mappings from variables to sets of values (namely ρ' and the greatest solution σ), *i.e.*, valuations of $\mathcal{D}^{Sets}$.

$$\rho' = \mathsf{greatestSolution}(\bigwedge_{p\in\mathsf{Proc}} \bar{x}_p = \bigcup_{i\in\{1,\ldots,n\}} \bar{x}_p^i \wedge \bigwedge_{i=1,\ldots,n} (\psi_i \wedge \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \rho(\bar{x}_{p_{ij}})))$$

We introduce the operator Φ over valuations of $\mathcal{D}^{Sets}$.

$$\Phi(\rho) = \mathsf{greatestSolution}(\bigwedge_{p\in\mathsf{Proc}} \bar{x}_p = \bigcup_{i\in\{1,\ldots,n\}} \bar{x}_p^i \wedge \bigwedge_{i=1,\ldots,n} (\psi_i \wedge \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \rho(\bar{x}_{p_{ij}})))$$

Let X_0 be the greatest fixpoint of the abstraction of the T_P operator. Then the representation ρ_0 of X_0 is the greatest solution of the following fixpoint equation.

$$\rho = \Phi(\rho)$$

We now show that $\rho_0 = \mathsf{greatestSolution}(\mathcal{K}_P^{\cup})$.

If ρ_1 is any fixpoint of Φ, then ρ_1 is the greatest solution of the conjunction $\mathcal{K}_1 \wedge \mathsf{K}_2$ of the set constraint

$$\mathcal{K}_1 = \bigwedge_{p\in\mathsf{Proc}} \bar{x}_p = \bigcup_{i\in\{1,\ldots,n\}} \bar{x}_p^i \wedge \bigwedge_{i=1,\ldots,n} \psi_i$$

and the formula

$$\mathsf{K}_2 = \bigwedge_{p\in\mathsf{Proc}} \bigwedge_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \rho_1(\bar{x}_{p_{ij}}).$$

Thus, ρ_1 is a solution also of $\mathcal{K}_P^{\cup} = \mathcal{K}_1 \wedge \mathcal{K}_2'$ where

$$\mathcal{K}_2' = \bigwedge_{p\in\mathsf{Proc}} \bigwedge_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \bar{x}_{p_{ij}}.$$

Thus, $\rho_1 \leq \mathsf{greatestSolution}(\mathcal{K}_P^{\cup})$ holds for any fixpoint of Φ, and in particular for the greatest one, namely ρ_0.

We next show that the greatest solution σ_0 of $\mathcal{K}_P^{\cup}$ is a fixpoint of Φ. If $\sigma_0 = \mathsf{greatestSolution}(\mathcal{K}_P^{\cup})$, then σ_0 is the greatest solution of $\mathcal{K}_1 \wedge \mathsf{K}_2''$ where

$$\mathcal{K}_2'' = \bigwedge_{p\in\mathsf{Proc}} \bigwedge_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,m_i} \bar{y}_{ij} \subseteq \sigma_0(\bar{x}_{p_{ij}})$$

since, if σ_1 is any other solution of $\mathcal{K}_1 \wedge \mathcal{K}_2''$ then $\sigma_0 \cup \sigma_1$ is also a solution of $\mathcal{K}_P^{\cup}$, and, thus, $\sigma_0 \cup \sigma_1 \leq \sigma_0$. Thus, $\sigma_0 = \Phi(\sigma_0)$.

4 Erroneous states and greatest model semantics

We now turn to transition systems that are induced by constraint logic programs according to their abstract operational semantics. The states are the error state err or pairs consisting of the parallel composition of procedure calls and of the constraint store, *i.e.*,

$$\mathcal{S} = \{\langle \bigwedge_{i=1,\ldots,n} p_i(\bar{x}_i), \varphi\rangle \mid n \geq 0,\ \varphi \text{ is a satisfiable constraint}\} \cup \{\mathsf{err}\}.$$

We identify states modulo the AC1-laws for parallel composition of procedure calls in the first component and modulo logical equivalence of the constraint stores in the second component. We use the same symbol $\wedge$ also for the logical 'and' composition of constraints; in both cases, $\top$ (*true*) is the neutral element; it corresponds either to the empty parallel composition or to the empty parallel conjunction.

The transition function $\mathcal{T}_P$ of the system induced by the program P is defined as follows. Let s be a state is of the form $s = \langle p(\bar{x}) \wedge E, \varphi\rangle$ and the procedure p be defined by n clauses as in the form above. , For any $i = 1, \ldots, n$, there is a transition from s into the state $s' = \langle \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}) \wedge E, \varphi \wedge \psi_i\rangle$,

$$\langle p(\bar{x}) \wedge E, \varphi\rangle \quad \longrightarrow_{\mathcal{T}_P} \quad \langle \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}) \wedge E, \varphi \wedge \psi_i\rangle$$

if $\varphi \wedge \psi_i$ is satisfiable. If no such transition exists (for that predicate p), then there is a transition from s into the error state err. An execution is *fair* if for every state s, every procedure p occurring in s will be applied (ie, yield a transition as described above) eventually (*i.e.*, for some state in the execution sequence starting in s).

For a given state $s = \langle p(\bar{x}), \varphi\rangle$, we define that $\lfloor s \rfloor_{\mathcal{D}} = \{p(\bar{d}) \mid \bar{d} \text{ is a solution of } \varphi\}$ (*i.e.*, the set of all ground instances of the query $p(\bar{x}) \wedge \varphi$). We extend this definition to sets of states in the canonical way.

Theorem 3 If the constraint system has the saturation property, then a state s terminates in the error state in every fair execution of the transition system induced by the program P if and only if none its ground instances lies in the greatest model of P (or, equivalently, in the greatest fixed point of T_P); *i.e.*,

$$\text{every execution starting in } \langle p(\bar{x}), \varphi\rangle \text{ leads to } \mathsf{err} \text{ iff } \lfloor \langle p(\bar{x}), \varphi\rangle \rfloor_{\mathcal{D}} \cap gfp(T_P) = \emptyset$$

Proof. Palmgren (1986) has shown that every program over a constraint domain with the saturation property is canonical. Therefore, by classical results, every *ground execution* starting in any ground instance $p(\bar{d})$ of $\langle p(\bar{x}), \varphi\rangle$ leads to err if and only if $p(\bar{d}) \notin gfp(T_P)$; *i.e.*,

$$GFF_P = \mathcal{B}_D - gfp(T_P).$$

Thus, we only need to show that every ground execution starting in any ground instance $p(\bar{d})$ of $\langle p(\bar{x}), \varphi\rangle$ leads to err if and only if every execution starting in $\langle p(\bar{x}), \varphi\rangle$ leads to err; *i.e.*,

$$\langle p(\bar{x}), \varphi\rangle \in FF_P \text{ iff } \lfloor \langle p(\bar{x}), \varphi\rangle \rfloor_{\mathcal{D}} \subseteq GFF_P.$$

The "only if" direction is clear. For the "if" direction, assume that there exists an execution starting in the state $s = \langle p(\bar{x}), \varphi\rangle$ that does not lead to err. That is, there exists a transition sequence $s = s_0, s_1, s_2, \ldots$ such that the constraint store, say φ_i of every state s_i is satisfiable. Since φ_i is stronger than φ_{i-1} for $i \geq 1$, this means that $\bigcap_{i=0,\ldots,n} \mathsf{Sol}(\varphi_i) \neq \emptyset$. The saturation property yields that also $\bigcap_{i\geq 0} \mathsf{Sol}(\varphi_i) \neq \emptyset$. Let α be a member in this set, *i.e.*, a solution of the infinite set constraint stores φ_i. Then the transition sequence $s'_o, s'_1, s'_2, \ldots$ that we obtain by instantiating the constraint store in each state s_i by the valuation α is a ground derivation that does not lead to err. Hence, if $\alpha(\bar{x}) = \bar{d}$, then $p(\bar{d}) \in \lfloor \langle p(\bar{x}), \varphi\rangle \rfloor_{\mathcal{D}} - GFF_P$. This completes the proof of the "if" direction.

5 The constraint domain of infinite trees

For the canonical extension of the constraint domain Herbrand (the constraints are equations between terms interpreted over finite trees) to set constraints with union, the interpretation of constructor symbols is the canonical extension from trees to sets of trees; inclusion and union have their usual set-theoretic interpretation. The first condition is satisfied, but the second one is not. Both conditions are satisfied if we interpret the inclusion relation as follows (the same holds true for the cases of rational and infinite trees).

$$\begin{array}{lll} \mathcal{D}^{Sets}, \sigma \models x \subseteq y & \text{iff} & \mathsf{Paths}(\sigma(x)) \subseteq \mathsf{Paths}(\sigma(y)) \\ \mathcal{D}^{Sets}, \sigma \models f(u_1, \ldots, u_n) \subseteq 0 & \text{iff} & \sigma(u_1) = \ldots = \sigma(u_n) = \emptyset \end{array}$$

The first condition corresponds to taking the usual "Cartesian" approximation of union. We define the set $\mathsf{Paths}(t)$ for a tree t (its extension to sets of trees is the canonical one) as the set of the path labelings "together with their direction".

$$\begin{array}{rcl} paths(a) & = & \{\langle a, \varepsilon\rangle\} \\ paths(f(t_1, \ldots, t_n)) & = & \{\langle f, 1\rangle.p \mid p \in paths(t_1)\} \cup \ldots \cup \{\langle f, n\rangle.p \mid p \in paths(t_n)\} \end{array}$$

The second condition accounts for the special role of the empty set (and corresponds to optimizations in existing implementations of set-constraint solvers).

6 Temporal properties of CLP systems

Let $(\mathcal{S}, \mathcal{T})$ be a transition system, *i.e.*, a set $\mathcal{S}$ of states together with a (non-deterministic) transition function $\mathcal{T} : \mathcal{S} \to \mathcal{P}(\mathcal{S})$, *i.e.*, a mapping from $\mathcal{S}$ to the powerset of $\mathcal{S}$. Given a property of states, *i.e.*, a set $\mathsf{Prop} \subseteq \mathcal{S}$, we can define a new set of states satisfying a CTL-style temporal property in one of the following ways.

$$\begin{array}{l} \mathsf{EF}(\mathsf{Prop}) = lfp(\lambda X.\ \mathsf{Prop} \cup \mathcal{T}^{-1}(X)) \\ \mathsf{EG}(\mathsf{Prop}) = gfp(\lambda X.\ \mathsf{Prop} \cap \mathcal{T}^{-1}(X)) \\ \mathsf{AF}(\mathsf{Prop}) = \mathcal{S} - \mathsf{EG}(\mathcal{S} - \mathsf{Prop}) \\ \mathsf{AG}(\mathsf{Prop}) = \mathcal{S} - \mathsf{EF}(\mathcal{S} - \mathsf{Prop}) \end{array}$$

Here, $\mathsf{EF}(\mathsf{Prop})$ ("exists finally Prop") denotes the set of all states $s \in \mathcal{S}$ for which there exists an execution starting in s and reaching a state s' in Prop, *i.e.*, there exists a sequence of states $s = s_0, \ldots, s_n = s'$ such that $s_i \in \mathcal{T}(\{s_{i-1}\})$ for $i = 1, \ldots, n$.

$$\mathsf{EF}(\mathsf{Prop}) = \bigcup_{i \geq 0} (\mathcal{T}^{-1})^i(\mathsf{Prop})$$

Similarly, $\mathsf{EG}(\mathsf{Prop})$ ("exists globally Prop") denotes the set of all states $s \in \mathcal{S}$ for which there exists an execution starting in s such that all reached states lie in Prop, *i.e.*, a (necessarily infinite) sequence of states $s = s_0, s_1, s_2, \ldots \in \mathsf{Prop}$ such that $s_i \in \mathcal{T}(\{s_{i-1}\})$ for $i = 1, \ldots, n$.

$$\mathsf{EG}(\mathsf{Prop}) = \bigcap_{i \geq 0} (\mathcal{T}^{-1})^i(\mathsf{Prop})$$

The meaning of the other two sets is obtained in the analogous way by referring to *all* executions starting in the states they contain.

The transition systems we consider are induced by (generally non-terminating) *constraint logic programs*. (They will also appear as approximations of those that are induced by concurrent constraint programs, introduced in the next section).

The following schema for specifying state properties is inspired by the abstract debugging framework of Bourdoncle (1991). We annote each predicate $p(\bar{x})$ with an *assertion* $\gamma(\bar{x})$ on the set of the possible values of $\bar{x}$. (One may fix default assertions, for example *true* or *false*.) Given such an annotation with assertions, we define:

$$\mathsf{Prop} = \{\langle \bigwedge_{i=1,\ldots,n} p_i(\bar{x}_i), \varphi\rangle \mid \bigwedge_{i=1,\ldots,n} \gamma_i(\bar{x}_i) \wedge \varphi \text{ is satisfiable}\}$$

Properties Prop of states defined in this way are invariant under parallel composition of procedure calls in the following sense. If s and s' lie in Prop then also the state that is the component-wise $\wedge$-composition of s and s'. We call a set of states thus defined an *invariant property*. We note that the above definition is one possibility to define properties of states without a bound on the possible number of procedure calls ("processes").

From now on, we take the definitions of temporal properties with respect to the transition function $\mathcal{T}$ induced by a corresponding CLP program. We define the following transformers of sets of ground facts.

$$T_{P\vee\mathsf{Prop}} = \lambda X.\ \lfloor\mathsf{Prop}\rfloor_{\mathcal{D}} \cup T_P(X)$$
$$T_{P\wedge\mathsf{Prop}} = \lambda X.\ \lfloor\mathsf{Prop}\rfloor_{\mathcal{D}} \cap T_P(X)$$

Our notation is justified by the fact that $T_{P\vee\mathsf{Prop}}$ is the operator associated with the program $P \vee \mathsf{Prop}$ below. We obtain this program from P by adding a disjunct of the form $\bar{x}_p \in \{\bar{d} \mid p(\bar{d}) \in \mathsf{Prop}\}$ to each definition of a predicate $p(\bar{x})$.

$$\begin{array}{rcl} P \vee \mathsf{Prop} & \equiv & \bigwedge_{p\in\mathsf{Proc}}\ p(\bar{x}_p) \leftrightarrow \bigvee_{i=1,\ldots n} (\bar{x}_p = \bar{x}^i_p \wedge \psi_i \wedge \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij})) \\ & & \vee\ \bar{x}_p \in \{\bar{d} \mid p(\bar{d}) \in \mathsf{Prop}\} \end{array}$$

In the analogous way, $T_{P\wedge\mathsf{Prop}}$ is the operator associated with the program $P \wedge \mathsf{Prop}$ below which we obtain by adding a conjunct of the same form to each clause in the definition of a predicate $p(\bar{x})$.

$$\begin{array}{rcl} P \wedge \mathsf{Prop} & \equiv & \bigwedge_{p\in\mathsf{Proc}}\ p(\bar{x}_p) \leftrightarrow \bigvee_{i=1,\ldots n} (\bar{x}_p = \bar{x}^i_p\ \wedge \psi_i \wedge \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}) \\ & & \wedge\ \bar{x}_p \in \{\bar{d} \mid p(\bar{d}) \in \mathsf{Prop}\}) \end{array}$$

If the conjunct can be expressed by a constraint ψ, then the programs $P \vee \mathsf{Prop}$ and $P \wedge \mathsf{Prop}$ are again pure CLP programs. In the framework where we derive set constraints, we may define the set Prop as the greatest solutions of a set constraint in $\mathcal{L}^{\cup}$ in programs $P \vee \mathsf{Prop}$, and as the least solution of set constraints in $\mathcal{L}^{\cap}$ (*i.e.*, with intersection) in programs $P \wedge \mathsf{Prop}$. [2]

Theorem 4 Given an invariant set of states Prop, there exists at least one execution starting in the state $s = \langle p(\bar{x}), \varphi\rangle$ and reaching a state in Prop if and only if there exists a solution of φ mapping $\bar{x}$ to $\bar{d}$ such that $p(\bar{d})$ lies in the least fixpoint of the operator $T_{P\vee\mathsf{Prop}}$.

$$\langle p(\bar{x}), \varphi\rangle \in \mathsf{EF}(\mathsf{Prop}) \quad\leftrightarrow\quad \lfloor\langle p(\bar{x}), \varphi\rangle\rfloor_{\mathcal{D}} \cap lfp(T_{P\vee\mathsf{Prop}}) \neq \emptyset$$

If the constraint domain $\mathcal{D}$ has the saturation property, there exists at least one execution starting in the state $s = \langle p(\bar{x}), \varphi\rangle$ such that every state reached during the execution lies in the set Prop if and only if there exists a solution of φ mapping $\bar{x}$ to $\bar{d}$ such that $p(\bar{d})$ lies in the greatest fixpoint of the operator $T_{P\wedge\mathsf{Prop}}$.

$$\langle p(\bar{x}), \varphi\rangle \in \mathsf{EG}(\mathsf{Prop}) \quad\leftrightarrow\quad \lfloor\langle p(\bar{x}), \varphi\rangle\rfloor_{\mathcal{D}} \cap gfp(T_{P\wedge\mathsf{Prop}}) \neq \emptyset$$

[2] In a more powerful framework, Prop may be defined by temporal properties itself; *i.e.*, one may nest temporal operators. One may obtain then set constraints with conjuncts of the form $x \subseteq \mathsf{leastSolution}(K^{\cap}_P)$ or $\mathsf{greatestSolution}(K^{\cup}_P) \subseteq x$. It is an open problem whether such set constraints are effectively solvable, and what their expressiveness is with respect to Rabin tree automata.

7 Concurrent Constraint Programs

We will now consider the transition systems that are induced by *concurrent constraint programs*. As before, its states are the error state or pairs consisting of the parallel composition of procedure calls and of the constraint store, *i.e.*,

$$\mathcal{S} = \{\langle \bigwedge_{i=1,\ldots,n} p_i(\bar{x}_i), \varphi\rangle \mid n \geq 0,\ \varphi \text{ is a satisfiable constraint}\} \cup \{\mathsf{err}\}.$$

We identify states modulo the same laws as before.

The transition function $\mathcal{T}_P$ of the system induced by the cc program P is defined according to the semantics of the "pure" cc programming constructs. The procedure p is defined either as the "tell" of a constraint ψ, *i.e.*,

$$p(\bar{x}) \leftrightarrow \exists_{-\bar{x}}\ \psi$$

or as the *committed choice* of several guarded clauses,

$$p(\bar{x}) \leftrightarrow \bigvee_{i=1,\ldots n} \exists X_i.\ \psi_i \parallel \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}).$$

In the first case, there is a transition from a state s of the form $s = \langle p(\bar{x}) \wedge E, \varphi\rangle$. into the state $s' = \langle E, \varphi \wedge \psi\rangle$ if $\varphi \wedge \psi$ are satisfiable, and into the error state err otherwise. In the second case, a transition going from $s = \langle p(\bar{x}) \wedge E, \varphi\rangle$ into the state $s' = \langle \bigwedge_{j=1,\ldots,m_i} p_{ij}(\bar{y}_{ij}) \wedge E, \varphi \wedge \psi\rangle$ is possible for any $i = 1, \ldots, n$ under the condition that φ entails $\exists X_i.\ \psi$.

If a state is of the form $s = \langle \top, \varphi\rangle$ (where $\top$ is the "empty" parallel composition of procedure calls; *i.e.*, the execution of the program terminates in s) then, for formal reasons, we require that there is a transition of every such state upon itself. The same holds for the error state.

We obtain upper approximations of the sets EF(Prop) and EG(Prop) and lower approximations of the sets AF(Prop) and AG(Prop) as follows. We relax the condition under which a clause in the committed-choice definition of a procedure $p(\bar{x})$ may be chosen (namely, the constraint store φ entails the guard ψ) to the condition that the conjunction of φ and ψ is satisfiable.

The approximated transition function corresponds to the abstract operational semantics of the constraint logic programming (CLP) paradigm. In that paradigm, however, executions of a program have a meaning only if they terminate. In this paper we are interested in generally non-terminating concurrent systems (which is why we use temporal properties as specifications). The approximated system does not account for synchronization based on suspension. It is still a concurrent system (in the sense, for example, of Lamport (1992)) in that its behavior is characterized by (interleaving) non-determinism plus fairness.