Constraint-based integrity checking in abductive and non-monotonic extensions of constraint logic programming

Aditya K. Ghose
Department of Business Systems
University of Wollongong,
NSW 2522 Australia
aditya@uow.edu.au

Srinivas Padmanabhuni
Department of Computing Science
University of Alberta, Edmonton
Alberta, Canada, T6G 2H1
srinivas@cs.ualberta.ca

Abstract

Recent research on the integration of the abductive and constraint logic programming paradigms has led to systems which are both expressive and computationally efficient. This paper investigates the role of constraints in integrity checking in the context of such systems. Providing support for constraints in this role leads to a framework that is significantly more expressive, without significant loss in efficiency. We augment the Abductive Constraint Logic Programming framework with *assumed constraints* and provide model- and proof-theoretic accounts of two variants: one which involves *commitment* to such assumptions, and one which does not. We also show that such accounts extend easily to a constraint logic programming framework which supports both negation and assumed constraints. The gains in expressivity in these frameworks turn out to be particularly useful in a variety of application domains, including scheduling and constraint database updates.

Introduction

Recent research on the integration of the abductive and constraint logic programming paradigms has led to systems which are both expressive and computationally efficient. The Abductive Constraint Logic Programming (ACLP) paradigm (Kakas & Michael 1995) involves the cooperation of abductive and constraint solvers in efficiently generating solutions to complex problems represented in an expressive, high-level language. This paper seeks to enhance the expressive power of frameworks such as ACLP by augmenting the representation language with constraints whose sole purpose is integrity checking. Equivalently, this may be thought of as the extension of the representation language with *assumed constraints*. Our intent is to develop more robust and expressive representational frameworks without sacrificing computational efficiency.

Integrity constraints play a key role in a variety of abductive and nonmonotonic reasoning systems. Reiter has convincingly argued in (Reiter 1988) that integrity constraints are best viewed as meta-theoretic assertions concerning the content of a knowledge base as opposed to object-level assertions in the knowledge base. To our knowledge, there has been no previous work on the use of constraints for integrity checking, either in the context of pure constraint-based reasoning, or hybrid frameworks such as constraint logic programming (CLP) (Jaffar & M.J.Maher 1994) or ACLP which integrate constraints with other reasoning systems (however, default reasoning notions have been considered in the context of concurrent constraint programming in (Saraswat, Jagadeesan, & Gupta 1996)). Yet, the question is a non-trivial one, since constraints used for integrity checking must meta-theoretically restrict the set of consequences of a knowledge base without restricting the values that solution variables may take. The following example clarifies the point. Let $\{x < 10, x > 5, y = 1, x + y < 9\}$ be a given set of constraints on the domain of integers. One may then specify $x \geq 8$ as an integrity constraint with the intention of ensuring that every set of constraints used to obtain a solution is consistent with it. The initial set of constraints is not consistent with this integrity constraint, but the following two subsets are:

- $\{x < 10, x > 5, y = 1\}$
- $\{x < 10, x > 5, x + y < 9\}$

Notice that both of these sets of constraints, while being consistent with $x \geq 8$, admit solutions which violate it (e.g., $x = 7, y = 1$ for the first set). Here the integrity constraint has been used to restrict the subproblem that is solved, without being enforced on the values that output variables may take in solutions. Thus, the integrity constraint plays precisely the same role as integrity constraints in abductive systems such as THEORIST (Poole, Goebel, & Aleliunas 1987). A THEORIST system with a and $a \rightarrow b$ as hypotheses and $\neg b$ as an integrity constraint admits two distinct maximal scenarios, containing a and $a \rightarrow b$ respectively. Notice that $\neg b$ is used to restrict what may appear in a maximal scenario, but does not itself appear in one. We argue that several classes of applications require that constraints predicates play such a role. No existing system supports this. For instance, constraint predicates that appear in the integrity constraints of ACLP are actually enforced on the output values that variables may take.

An alternative view of constraints in integrity checking roles treats them as *assumptions*, in the same sense

as the justifications of default rules in default logic (Reiter 1980) (which are used as assumptions concerning the content of extensions with the consistency of such assumptions sanctioning the application of a default rule). Our intent is to explore how *assumed constraints* might play a role similar to default justifications in hybrid frameworks such as ACLP and CLP by serving as filters that must be checked to determine the applicability of a program clause. Once again, constraint assumptions restrict the set of valid conclusions but are not themselves derivable as conclusions. In the context of systems such as ACLP and CLP, we distinguish between two classes of constraints: *enforced constraints* (which constrain the values that output variables may take) and *assumed constraints* in the sense just discussed.

Constraints as assumptions are useful in variety of domains including planning, scheduling and configuration. The following is a typical rule that one might need to encode in a scheduling application:

> *If it is consistent to assume that a given order is ready to be shipped and that the order arrives at the loading dock at time t1 and that truck A may be loaded at time t2, then conclude that truck A must be loaded at time t2, provided the constraint t2=t1+5 is satisfiable.*

Notice that $t2 = t1+5$ is not an *enforced constraint* since the values of $t1$ and $t2$ are independently determined. It is an *assumed constraint* which, if violated, blocks the applicability of the corresponding rule.

Constraints used for integrity checking find useful application in enforcing constraint database updates. Returning to our initial example involving a set of constraints on the domain of integers, the retraction of the constraint $x < 8$ from a database given by $\{x < 10, x > 5, y = 1, x + y < 9\}$ is achieved simply by asserting $x \geq 8$ as an integrity constraint. Similarly, one might wish to specify constraints which determine valid updates of the values of certain variables, without enforcing them on the values of output variables in solutions. In a companion paper, we describe how constraint predicates in integrity checking roles may be used to define a meaningful notion of *minimal specialization* in formalizations of incremental learning of constraint-based representations. One such account of incremental learning is being trialed in a system for automatically acquiring scheduling knowledge. Constraints in integrity checking roles may be used for enforcing the mutual exclusivity of constraints. Thus, the mutual exclusivity of constraints c_1 and c_2 may be enforced by asserting $\neg(c_1 \wedge c_2)$ as an integrity constraint without requiring that the resulting constraint be enforced on solutions. Constraint assumptions may be used to implement a weak notion of conditional constraints (e.g., *enforce* c_1 *if it is consistent to assume* c_2).

We discuss two variants of the ACLP framework in detail in this paper. Both involve augmenting the basic ACLP representational framework by permitting the specification of *assumed constraints* in both the program clauses and the integrity constraints. In the first variant, which we shall call the *Extended ACLP (E-ACLP) framework*, assumed constraints in program clauses are tested for consistency against a *constraint store* consisting of the current set of accumulated enforced constraints to determine whether the corresponding clause is applicable. The second variant, which we shall call the *EC-ACLP framework*, involves *committing to assumptions* in a manner analogous to what Rational Default Logic (Mikitiuk & Truszczynski 1993) and Constrained Default Logic (Delgrande, Schaub, & Jackson 1994) achieve in the context of default reasoning. The intent is to ensure that contradictory constraint assumptions are not made; this is achieved by maintaining a separate store which accumulates all assumed constraints in addition to the enforced ones. We present a model theory and proof theory for each variant. We then show how these semantics and proof procedures extend to the case of CLP with default negation (applicable to non-constraint predicates) and assumed constraints.

Extending ACLP with constraints for integrity checking

In this section we shall define the syntax and present the model theory of the E-ACLP and EC-ACLP frameworks. As with CLP, we shall assume that both these frameworks are parameterized by a choice of constraint domain $\mathcal{D}$. We shall use the standard notions of *entailment* in constraint domain, and *valuations* as defined in (Jaffar & M.J.Maher 1994). In the following, B_P refers to the Herbrand base of P.

Extended ACLP

Definition 1 [Extended ACLP Framework] *An Extended ACLP($\mathcal{D}$) framework is a triple (P, A, IC) where:*

- *P is a collection of clauses of the form:*
  $p \leftarrow q_1, \ldots, q_m[]c_1, \ldots, c_n : d_1, \ldots, d_r$
  where $p, q_1, \ldots, q_m$ are non-constraint predicate symbols while $c_1, \ldots, c_n$ and $d_1, \ldots, d_r$ are constraint predicates defined on $\mathcal{D}$.
- *A is a distinguished subset of the set of non-constraint predicates called* abducibles.
- *IC, the set of* integrity constraints *is a collection of clauses of the form:*
  $\bot \leftarrow q_1, \ldots, q_m[]c_1, \ldots, c_n : d_1, \ldots, d_r$
  where $q_1, \ldots, q_m$ are non-constraint predicate symbols while $c_1, \ldots, c_n$ and $d_1, \ldots, d_r$ are constraint predicates defined on $\mathcal{D}$.

Program clauses of the form $p \leftarrow q_1, \ldots, q_m[]c_1, \ldots, c_n : d_1, \ldots, d_r$ read as follows: *if $q_1, \ldots, q_m$ have been established and the set of all enforced constraints c_i together with all assumed*

constraints d_i are consistent with the current constraint store of accumulated enforced constraints, conclude p and add each c_i to the constraint store. An integrity constraint of the form $\bot \leftarrow q_1, \ldots, q_m [] c_1, \ldots, c_n : d_1, \ldots, d_r$ ensures that the following are not simultaneously true: $q_1, \ldots, q_m$ are simultaneously derivable, each enforced constraint c_i is entailed by the constraint store and the constraint store is consistent with the assumed constraints d_i.

Definition 2 *Let P be an E-ACLP program. For any pair (R,T) where $R \subseteq B_P$ and T is any set of constraint predicates on the domain $\mathcal{D}$, let $\Gamma(R,T) = (R',T')$ where R' and T' are the smallest sets such that:*

- $R' \subseteq B_P$
- *T' is a set of constraint predicates defined on $\mathcal{D}$.*
- *for any clause $p \leftarrow q_1, \ldots, q_m [] c_1, \ldots, c_n : d_1, \ldots, d_r \in P$, if there exist valuations v and v' such that $v(q_1), \ldots, v(q_m) \in R$, $\mathcal{D} \models v(c_1) \wedge \ldots \wedge v(c_n) \wedge v(T)$ and $\mathcal{D} \models v'(c_1) \wedge \ldots \wedge v'(c_n) \wedge v'(d_1) \wedge \ldots \wedge v'(d_r) \wedge v'(T)$, then $v(p) \in R'$, $\{c_1, \ldots, c_n\} \subseteq T'$*

A pair (M,S), where $M \subseteq B_P$ and S is a set of constraints defined on $\mathcal{D}$, is a model of P iff $\Gamma(M,S) = (M,S)$ and $\bot \notin M$.

Definition 3 [E-ACLP Abductive Explanations] *Given an E-ACLP framework (P,A,IC), an explanation of a goal of the form:*

$$q_1, \ldots, q_m [] c_1, \ldots, c_n : d_1, \ldots, d_r$$

is a set Δ of clauses of the form:

$$a \leftarrow e_1, \ldots, e_s$$

where a is an atom whose predicate symbol belongs to A and each e_i is a constraint predicate defined on $\mathcal{D}$, such that there exists a model (M,S) of $P \cup \Delta \cup IC$ such that $S \cup \{c_1, \ldots, c_n\} \cup \{d_1, \ldots, d_r\}$ is solvable and for every valuation v where $\mathcal{D} \models v(S \cup \{c_1, \ldots, c_n\})$, $\{v(q_1), \ldots, v(q_m)\} \subseteq M$.

Adding commitment to assumptions in extended ACLP

Syntactically, EC-ACLP frameworks are identical to E-ACLP frameworks. However, the notions of model and abductive explanation differ. EC-ACLP models augment E-ACLP models by explicitly recording every constraint assumption made (in addition to all enforced constraint) in a separate store. This store thus enables explicit commitment to assumptions. For a program clause to be applicable, its assumed constraints and its enforced constraints must be jointly consistent with this store. If a program clause is deemed to be applicable, all assumed and enforced constraints contained in the clause are added to this store (in addition to the enforced constraints being added to the set of accumulated enforced constraints). Each integrity constraint involves a similar consistency check with this additional constraint store.

Definition 4 *Let P be an EC-ACLP program. For any triple (R,S,T) where $R \subseteq B_P$ while S and T are sets of constraint predicates on the domain $\mathcal{D}$, let $\Gamma(R,S,T) = (R',S',T')$ be the smallest set R', S' and T' such that:*

- *$R' \subseteq B_P$*
- *S' and T' are constraint predicates defined on $\mathcal{D}$.*
- *for any clause $p \leftarrow q_1, \ldots, q_m [] c_1, \ldots, c_n : d_1, \ldots, d_r \in P$, if there exist valuations v and v' such that $v(q_1), \ldots, v(q_m) \in R$, $\mathcal{D} \models v(c_1) \wedge \ldots \wedge v(c_n) \wedge v(T)$ and $\mathcal{D} \models v'(c_1) \wedge \ldots \wedge v'(c_n) \wedge v'(d_1) \wedge \ldots \wedge v'(d_r) \wedge v'(S)$, then $v(p) \in R'$, $\{c_1, \ldots, c_n\} \subseteq T'$ and $\{c_1, \ldots, c_n\} \cup \{d_1, \ldots, d_r\} \subseteq S'$.*

A triple (M,C,S), where $M \subseteq B_P$ and C and S are constraints defined on $\mathcal{D}$, is a model of P iff $\Gamma(M,C,S) = (M,C,S)$ and $\bot \notin M$.

Definition 5 [EC-ACLP Abductive Explanations] *Given an EC-ACLP framework (P,A,IC), an explanation of a goal of the form:*

$$q_1, \ldots, q_m [] c_1, \ldots, c_n : d_1, \ldots, d_r$$

is a set Δ of clauses of the form:

$$a \leftarrow e_1, \ldots, e_s$$

where a is an atom whose predicate symbol belongs to A and each e_i is a constraint predicate defined on $\mathcal{D}$, such that there exists a model (M,C,S) of $P \cup \Delta \cup IC$ such that $S \cup \{c_1, \ldots, c_n\} \cup \{d_1, \ldots, d_r\}$ is solvable and for every valuation v where $\mathcal{D} \models v(C \cup \{c_1, \ldots, c_n\})$, $\{v(q_1), \ldots, v(q_m)\} \subseteq M$.

Observation: The E-ACLP and EC-ACLP frameworks without assumed constraints in the program clauses and integrity constraints coincide with the ACLP framework.

Proof Procedure for EC-ACLP

In this section, we present a proof procedure for EC-ACLP which builds on the abductive proof procedure defined in (Kakas & Michael 1995). For convenience let us denote a clause R_j of the above form by the short form $p \leftarrow Q_j [] C_j : D_j$ where C_j denotes the set of *enforced* constraints in the clause and D_j denotes the set of *assumed* constraints in the clause and Q_j represents the predicate literals in the clause R_j. A solution to the set of clauses of the above form are of the form $\exists x (a(x), C(x))$ where C(x) is a set of *enforced* constraints on the variables in the vector x in the abducible predicate. The framework is a direct generalization of the concept of clauses in CLP(R). The only additional feature in the clauses of EC-ACLP($\mathcal{D}$) is the presence of the *assumed* constraints. The *assumed* constraints of the form d_i are responsible for constraining the solution space of the solutions to the constraint logic program. Without loss of generality we shall assume any goal clause to be of the form $\leftarrow L_1, L_2, \ldots, L_n [] c_1, \ldots, c_m$, because corresponding to any generic goal of the form $\leftarrow L_1, L_2, \ldots, L_n [] c_1, c_2, c_3, \ldots, c_m : d_1, d_2, \ldots, d_k$, we can add the integrity constraints $\leftarrow L_1, L_2, \ldots, L_n [] \phi :$

$\neg d_1.,\ldots, \leftarrow L_1, L_2, \ldots, L_n[]\phi : \neg d_k$. (where $\neg d_1, \neg d_2 etc$ denote the classical negation of the constraints $d_1, d_2 \ldots$ respectively), to the set of integrity constraints IC, and pass the initial goal $\leftarrow L_1, L_2, \ldots, L_n[]c_1, c_2, c_3, \ldots, c_m$ as the initial goal G_1 to the abductive proof procedure, still preserving the semantics.

Abductive Derivation: An abductive derivation from $(G_1, \delta_1, \delta_1^*, J_1)$ to $(G_k, \delta_k, \delta_k^*, J_k)$ in $< P, A, IC >$ is a sequence $(G_1, \delta_1, \delta_1^*, J_1)$, $(G_2, \delta_2, \delta_2^*, J_2), \ldots$,$(G_k, \delta_k, \delta_k^*, J_k)$ such that for each i, G_i has the form $\leftarrow L_1, L_2, \ldots, L_n[]C$, C being a set of *enforced* constraints(could be empty), L_j is a selected atom, each δ_j is a set of ground abducibles over the domain of R extended by the skolem constants, each δ_j^* is a set of goals , J_i is a set of *enforced* and *assumed* constraint terms denoting the set of constraints of both kind encountered in the present goal G_i and $(G_{i+1}, \delta_{i+1}, \delta_{i+1}^*, J_{i+1})$ is derived according to the following rules:

A1) If L_i is not an abducible then $G_{i+1} = S$, $\delta_{i+1} = \delta_i$, J_{i+1}=$J_i \cup C_j \cup D_j$ and $\delta_{i+1}^* = \delta_i^*$ where S is the set of *enforced* constraints and literals in the CLP resolvent of some clause R_j of the form $p(u) \leftarrow Q_j[]C_j : D_j$ in P with G_i on L_j and the set of constraints J_{i+1} is solvable.

A2) If L_j is an atom with an abducible predicate and L_j unifies with a member of δ_i returning the constraints C_θ and $J_i \cup C_\theta$ is solvable then $G_{i+1} = \leftarrow, \ldots, L_{j-1}, L_{j+1}, \ldots, L_n[]C \cup C_\theta$, $\delta_{i+1} = \delta_i$ and $\delta_{i+1}^* = \delta_i^*$, $J_{i+1} = J_i \cup C_\theta$.

A3) If L_j is an atom with an abducible predicate and $L_j \notin \delta_i$, let $Sk(L_j) = L_j\theta = L_j^{'}$ and $C^{''} = C \cup \{x = t \mid x \in var(L_j), t = x\theta\}$. $J^{''} = J_i \cup \{x = t \mid x \in var(L_j), t = x\theta\}$. Then there exists a consistency derivation from $(\{L_j^{'}\}, C^{''}, \delta_i \cup \{L_j^{'}\}, \delta_i^*, J^{''})$ to $(\{\}, C^{'}, \delta^{'}, \delta^{*'}, J^{'})$ then $G_{i+1} = \leftarrow L_1, \ldots, L_3, L_4, \ldots, L_n[]C^{'}$. $\delta_{i+1} = \delta^{'}$ and $\delta_{i+1}^* = \delta^{*'}$ and $J_{i+1} = J^{'}$.

Consistency derivation A consistency derivation for an abducible atom L, from $(L, C_1, \delta_1, \delta_1^*, J_1)$ to $(F_m, C_m, \delta_m, \delta_m^*, J_m)$ in $< P, A, IC >$ is a sequence $(L, C_1, \delta_1, \delta_1^*, J_1)$,$(F_1, C_1, \delta_1, \delta_1^*, J_1)$,$(F_2, C_2, \delta_2, \delta_2^*, J_2)$, $\ldots$, $(F_m, C_m, \delta_m, \delta_m^*, J_m)$ where:

i) F_1 is the set of resolvents of the form G_{11},G_{12} etc. where all goals G_{1k} are of the form $\leftarrow L_1, \ldots, L_n[]C_{di} : D_{di}$ obtained by resolving the abducible L with the denials in IC and δ_1^*. All the constraints C_{di} here involved in any goal are *enforced* constraints, and D_{di} contain only *assumed* constraints.

ii) For each i > 1, F_i has the form $\{\leftarrow L_1, .., L_n[]C_{di} : D_{di}\} \cup F_i^{'}$, L_j or C_{di} or D_{di} is selected and $(F_{i+1}, C_{i+1}, \delta_{i+1}, \delta_{i+1}^*, J_{i+1})$ is obtained according to the following rules:

C1) If L_j is not an abducible, let R be the set of all r_i where r_i is a resolvent of $\leftarrow L_1, \ldots, L_n[]C_{di} : D_{di}$ with clauses in P on L_j and J_{ij} is the set of *enforced* as well as *assumed* constraints in the resolvent in conjunction with J_i, and J_{ij} is solvable. $F_{i+1} = R \cup F_i^{'}$ $C_{i+1} = C_i$ $\delta_{i+1}^* = \delta_i^*$ $\delta_{i+1} = \delta_i$. and $J_{i+1} = J_i$.

(C2) If L_j is an abducible predicate atom, let R be the set of all r_i where r_i is a resolvent of $\leftarrow L_1, \ldots, L_n[]C_{di} : D_{di}$ with atoms in δ_i on L_j and J_{ij} is the set of *enforced* as well as *assumed* constraints in the resolvent in conjunction with J_i, and J_{ij} is solvable. $F_{i+1} = R \cup F_i^{'}$ $C_{i+1} = C_i$ $\delta_{i+1}^* = \delta_i^* \cup \{\leftarrow L_1, \ldots, L_n[]C_{di} : D_{di}\}$ $\delta_{i+1} = \delta_i$ and $J_{i+1} = J_i$.

C3) if C_{di} is selected then $F_{i+1} = F_i^{'}$, $C_{i+1} = C_i \cup C^{'}$, $\delta_{i+1}^* = \delta_i^*$ and $\delta_{i+1} = \delta_i$ and $J_{i+1} = J_i \cup C^{'}$. Here $C^{'}$ is such that $J_i \cup C^{'}$ is solvable but $C_{di} \cup C^{'}$ is not solvable.

C4) if D_{di} is selected then $F_{i+1} = F_i^{'}$, $C_{i+1} = C_i$ $\delta_{i+1}^* = \delta_i^*$ and $\delta_{i+1} = \delta_i$ and $J_{i+1} = J_i \cup D^{'}$. Here $D^{'}$ is such that $J_i \cup D^{'}$ is solvable but $D_{di} \cup D^{'}$ is not solvable.

The difference from the treatment of D_{di} from C_{di} is that the constraint $D^{'}$ is only added to the justifications and not to the actual solution to enforce consistency, while in C_{di} $C^{'}$ is used to constrain the solution set of constraints in C_i.

Theorem 1 *If $< P, A, IC >$ is an EC-ACLP program, and $(\leftarrow G, \{\}, \{\}, \{\})$,..,$(\leftarrow \phi[]C, \delta, \delta^*, J)$ is an abductive derivation, it is termed as an abductive refutation. If $\delta_{sol} = Rs(\delta \cup \exists xC)$ where x is the vector of free variables in the constraint store C, and Rs stands for the reverse skolemisation function, then δ_{sol} is an abductive explanation for the goal G.*

Example 1 Consider the ACLP with *assumed* constraints in the following set of EC-ACLP clauses.

$C1 : s(X, Y) \leftarrow r(X), p(Y)[]X + Y < 9.$
$C2 : s(X, Y) \leftarrow r(X)[]X + Y > 8$
$C3 : p(Y) \leftarrow []Y = 1.$
$C4 : r(X) \leftarrow q(X), a(X), []X > 5, X < 10.$
$C5 : q(X) \leftarrow [] : X \geq 8.$

We have an integrity constraint IC1 as $\leftarrow a(X)[]X \leq 3$. It has no *assumed* constraint in the global IC. Consider the query $\leftarrow s(X, Y)[]\phi : \phi$. In the absence of any *assumed* or *enforced* constraint, we start with $G_1 = \leftarrow s(X, Y)[]\phi$. At the outset, we resolve this with the clause C1. The CLP resolvent is $\leftarrow r(X), p(X)$ and constraint store C= $\{X+Y < 9\}$, and J = $\{X+Y < 9\}$. On further resolving this with the clause C3, we get new goal as $\leftarrow r(X)$, with the constraint store C= $\{Y = 1, X + Y < 9\}$ and J = $\{Y = 1, X + Y < 9\}$. Now r(X) can be resolved with clause C4, to get $\leftarrow q(X), a(X)$ with the constraint store C and J both

as $\{X > 5, X < 10, Y = 1, X + Y < 9\}$. Next q(X) can be resolved with the clause C5, and here the *assumed* constraint $X \geq 8 \cup J$ is not solvable, hence this path of finding a solution stops. Retracting the path, we now try to resolve the goal $\leftarrow s(X,Y)$ with the clause C2. The CLP resolvent is $\leftarrow r(X)$, with the constraint store C $=\{X + Y > 8\}$, and J $= \{X + Y > 8\}$. On further resolving this with the clause C4, we get the new goal as $\leftarrow q(X), a(X)$, with C $= \{X > 5, X < 10, X + Y > 8\}$ and J = C. Next when we resolve q(X) with C5, we get the *assumed* constraint $X \geq 8$. This constraint is solvable in conjunction with J. So we can carry out the resolution and now J $= \{X > 5, X < 10, X + Y > 8, X \geq 8\}$. C still remains the same. Now since we only have the abducible a(X), we skolemise a(X), to a(t), t being a skolem constant. We add the substitution X=t to J as well as C. Now we need to obtain a consistency derivation for a(t), with $J_1 = \{X > 5, X < 10, X + Y > 8, X \geq 8, X = t\}$, $\delta_1 = \{a(t)\} C_1 = \{X > 5, X < 10, X + Y > 8, X = t\}$ $\delta_1^* = \{\}$.

The only integrity constraint against which a(t) can be resolved is the integrity constraint IC1. On resolving $\leftarrow a(t)$, against IC1, we get the resolvent $\leftarrow Y \leq 3, Y = t$ All others remain the same. Now to get the consistency derivation we note that there are no abducibles in this resolvent. Thus by selecting rule C3, we select $t > 3$ as the constraint $C^{'}$ such that $J_i = \{X > 5, X < 10, X + Y > 8, X \geq 8, X = t\} \cup C^{'}$ is solvable but $C^{'} \cup \{Y = t, Y \leq 3\}$ is not satisfiable. So we end the consistency derivation with $F_m = , C_m = \{X > 5, X < 10, X + Y > 8, X = t, t > 3\}$, $\delta_m = \{a(t\}$ and $\delta_m^* = \{\}$ and $J_m = \{X > 5, X < 10, X + Y > 8, X \geq 8, X = t, t > 3\}$.

So coming back to the abductive derivation, we get $G_{i+1} = \leftarrow X > 5, X < 10, X + Y > 8, X = t, t > 3$, where we have no more abducibles left. Hence the resulting solution for the query $\leftarrow s(X,Y)$ is the set of constraints $\{X > 5, X < 10, X + Y > 8, X = t, t > 3\}$, which when condensed along with δ_i and reverse skolemised gives $\leftarrow a(X)[] X > 5, X < 10, X + Y > 8$ as the solution.

Proof Procedure for E-ACLP

In the previous section, the detailed proof procedure for the EC-ACLP with commitment to assumptions was explored in detail. The semantics of E-ACLP without commitment to assumptions offers a more general view of obtaining solution explanations than the rigid framework offered by the commitment to assumptions. A simple modification of the procedure defined for the semantics in the previous section is sufficient to capture the notion of solution in this semantics. As already explained in the case of EC-ACLP, the *assumed* constraints can be presumed absent in the goal clause, without loss of generality.

Changes The following changes in the abductive and consistency derivations of the proof procedure shall achieve the required result.

1. In all the steps, whether the consistency derivation or the abductive derivation, the set of justifications J_i is no longer required.
2. In any step where the solvability of J_i is seen in conjunction with the set of *enforced* as well as *assumed* constraints in the resolvent, the new form requires solvability of the existing constraint store C_i of *enforced* constraints, in conjunction with the set of both *enforced* as well as the *assumed* constraints in the resolvent. The rules which are affected in this change are A1,A2 and C1,C2,C3 and C4.
3. In case, the set of constraints in C_i, is solvable in conjunction with the constraints of both types in the resolvent, add the *enforced* constraints in the resolvent to the constraint store, in rules A1 and A2.
4. The entire consistency derivation has no use of J_i in the steps. In step C3, only C_i is modified by addition of such a $C^{'}$. In step C4, do no changes to anything but only look for the existence of a $D^{'}$ such that $D^{'}$ is solvable in conjunction with C_i but not solvable with D_{di}.

Example of commitment-free E-ACLP Consider Example 1 in the previous section for the goal $\leftarrow s(X,Y)$ for the same set of clauses and integrity constraints. We start by resolving this with the clause C1. The CLP resolvent is $\leftarrow r(X), p(X)$ and constraint store C= $\{X + Y < 9\}$. On further resolving this with the clause C3, we get new goal as $\leftarrow r(X)$, with the constraint store C=$\{Y = 1, X + Y < 9\}$. r(X) can be resolved with clause C4, to get $\leftarrow q(X), a(X)$ with the constraint store C as $\{X > 5, X < 10, Y = 1, X + Y < 9\}$ and Next the q(X) can be resolved with the clause C5, and here the *assumed* constraint $X \geq 8 \cup$ C is not solvable, hence this path of finding a solution stops. Retracting the path, we now try to resolve the goal $\leftarrow s(X,Y)$ with the clause C2. The CLP resolvent is $\leftarrow r(X)$, with the constraint store C $= \{X + Y > 8\}$, On further resolving this with the clause C4, we get the new goal as $\leftarrow q(X), a(X)$ with C= $\{X > 5, X < 10, X + Y > 8\}$. Next when we resolve q(X) with C5, we get the *assumed* constraint $X \geq 8$. This constraint is solvable in conjunction with C. So we can carry out the resolution and C still remains the same. Now since we only have the abducible a(X), we skolemise a(X), to say a(t). We add the substitution X=t to C. Now we need to obtain a consistency derivation for a(t), with $\delta_1 = \{a(t)\} C_1 = \{X > 5, X < 10, X + Y > 8, X = t\}$. $\delta_1^* = \{\}$ The only integrity constraint against which a(t) can be resolved is the integrity constraint IC1. On resolving $\leftarrow a(t)$, against IC1, we get the resolvent $\leftarrow Y \leq 3, Y = t$ All others remain the same. Now to get the consistency derivation we note that there are no abducibles in this resolvent. Thus by selecting rule C3, we select $t > 3$ as the constraint $C^{'}$ such that $C_i = \{X > 5, X < 10, X + Y > 8, X = t\} \cup C^{'}$

is solvable but $C' \cup \{Y = t, Y \leq 3\}$ is not satisfiable. So we end the consistency derivation with $F_m = ,C_m = \{X > 5, X < 10, X + Y > 8, X = t, t > 3\}$, $\delta_m = \{a(t\}$ and $\delta^*_m = \{\}$. The final constraint set is $\leftarrow X > 5, X < 10, X + Y > 8, X = t, t > 3$ giving the final solution as $\leftarrow a(X)[]X > 5, X < 10, X + Y > 8$.

CLP extensions

In this section, we shall provide some comments on the semantics and proof procedures required for CLP programs with default negation (applicable to non-constraint predicates) and assumed constraints. We shall refer to these as CLPNC programs. The syntax of such programs is similar to that of program clauses in E-ACLP and EC-ACLP, except that each q_i is now additionally permitted to be of the form $notq(x_1, \ldots, x_k)$ as well. The model theory relies on the Gelfond-Lifschitz transform that forms the basis of stable model semantics (Gelfond & Lifschitz 1990). Let $M' \subseteq B_P$ for a CLPNC program P. Let P' represent the program obtained by applying every possible valid grounding to clauses in P. Let P'' be the program obtained by applying the Gelfond-Lifschitz transform to P', ignoring both enforced and assumed constraints. To obtain semantics similar to those for E-ACLP (i.e., without commitment to assumptions), we compute the model (M, S) obtained by treating P'' as an E-ACLP program. Then (M, S) is treated as the model of the original program P iff M coincides with M'. To obtain semantics similar to those for EC-ACLP (i.e., with commitment to assumptions), we compute the model (M, C, S) obtained by treating P'' as an EC-ACLP program. Then (M, C, S) is treated as the model of the original program P iff M coincides with M'.

In contrast to the model-theoretic semantics which computes total extensions in the form of stable models, the proof procedures involved in logic programs with negation formulate a top-down query answering procedure for a target goal. The Eshghi-Kowalski procedure (Eshghi & Kowalski 1989) has been modified for use in logic programming with negation by Dung (Dung 1995) by considering negation as failure as a kind of abduction. The scenarios(models) generated by this top down procedure have been related to stable models and other models of logic programs with negation in (Dung 1995). There have been specific versions of the Eshghi-Kowalski procedure proposed by Dung for negation as failure in (Dung 1995), with minor variations to take care of the specific integrity constraints present in case of negation by failure as the only rule of abduction.

Here too the only difference from the EC-ACLP framework above is the formulation of integrity constraints. The only integrity constraints IC in the CLPNC framework are of the form $\forall(t_i)p_i(t_i) \vee notp_i(t_i)$ and $\forall(t_i)\neg(p_i(t_i) \wedge notp_i(t_i))$. This pair of constraints holds for all predicates p_i and its corresponding abducible predicate $notp_i$. Here t_i's are vectors of variables contained in p_i. The only abducibles are the those of type $notp_i$.

Conclusions

We have examined in this paper a set of related frameworks which implement the notion of constraint assumptions, or constraints in integrity checking roles, motivated by the need to represent complex problems drawn from a broad range of application domains, including planning, scheduling and configuration. We have presented the model-theoretic and proof-theoretic bases for two distinct extensions of the ACLP framework. We have also outlined how these, coupled with intuitions from the stable model semantics of logic programs with negation, might be used to define a model theory and proof theory for CLP programs with default negation and assumed constraints. We believe these are important advances, yielding more robust and expressive systems without sacrificing computational efficiency.

References

Delgrande, J. P.; Schaub, T.; and Jackson, W. K. 1994. Alternative approach to default logic. *Artificial Intelligence* 70:167–237.

Dung, P. 1995. An argumentation theoretic foundation for logic programming. *J. Logic Programming* 22:151–177.

Eshghi, K., and Kowalski, R. 1989. Abduction compared with negation by failure. In *Proc. 6th ICLP*, 234–254. MIT Press.

Gelfond, M., and Lifschitz, V. 1990. Logical programs with classical negation. In *Proc. 7th ICLP*, 579–597. MIT Press.

Jaffar, J., and M.J.Maher. 1994. Constraint logic programming: a survey. *Journal of Logic Programming* 503–581.

Kakas, A., and Michael, A. 1995. Integrating abductive and constraint logic programming. In *Proc. 12th ICLP*, 399–413.

Mikitiuk, A., and Truszcynski, M. 1993. Rational default logic and disjunctive logic programming. In *Logic programming and nonmonotonic reasoning.* MIT Press.

Poole, D.; Goebel, R.; and Aleliunas, R. 1987. Theorist: A logical system for defaults and diagnosis. In Cercone, N., and McCalla, G., eds., *The Knowledge Frontier*. Springer. 331–352.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Reiter, R. 1988. On integrity constraints. In *Proc. of TARK-88*, 97–112.

Saraswat, V.; Jagadeesan, R.; and Gupta, V. 1996. Timed default concurrent constraint programming. *Journal of Symbolic Computation* 22(5-6):475–520.

Thomas Eiter, J. L., and Subrahmanian, V. 1997. Computing non-ground representations of stable models. In *Proc of 4th LPNMR*, 198–217. Springer-Verlag.