Using Continuous Domain Logic to Solve Interval Constraint Satisfaction Problems

Benjamín R. Callejas Bedregal and Benedito Melo Acióly
Universidade Federal do Rio Grande do Norte - UFRN
Departamento de Informática e Matemática Aplicada - DIMAp
Laboratório de Lógica e Inteligência Computacional - LabLIC
Campus Universitário - Lagoa Nova
CEP: 59.072-970, Natal-RN, Brasil.
Fone: (55)(84)215-3814 Fax: (55)(84)215-3781
bedregal@dimap.ufrn.br and bma@dimap.ufrn.br

Abstract

Domain Logic (algebraic) was introduzed by Abramsky [Abr91] in order to study the logical aspects (a proof system) of domains as used in the denotational semantics of programming languages. Thus, we present denotational semantics as a program logic. In [BA95] the Abramsky approach for domain logic was extended, interpreting types as continuous domains instead of algebraic domains.

In this work we extend the continuous domain logic incorporing the interval data type as primitive type and adding variables, interval arithmetic, axioms and rules to the interval data type. That allows us, for example, to solve interval constraint satisfaction problems as deduction in our logic.

Keywords: Domain Theory, Reals Interval, Domain Logic, Interval Constraint Satisfaction Problems.

1 Introduction

The (algebraic) domain logic was introduzed by Abramsky [Abr91] in order to study the logical aspects (a proof system) of domains as used in denotational semantics of programming languages. However, algebraic domains does not support a natural and topologically consistent interpretation of the real interval data type. Nevertheless, continuous domains have a desirable interpretation for this data type.

Analogously to the work of Scott [Sco82] to represent concretely algebraic domains, Hoofman in [Hoo93] introduzed the alternative description of Continuous Information Systems (CIS) for continuous domains. This representation makes explicit the idea of information, in the sense that each element of a domain must be seen as the collection of informations that "satisfy". Although of both notions are categorically equivalent, information systems allows us to capture the logical aspects of domains, in the sense that their properties can be derived from assumptions about the entailment between propositions expressing properties of computations. Thus, for example, the continuous domain of real interval is represented as a continuous information system whose basic informations are rational intervals.

In [BA95] the Abramsky approach to domain logic [Abr91] was extended interpreting types as continuous information systems instead of algebraic domains.

In this work, we will extend the typed languages introducing a distinguished type for representing the interval data type as well as formation rules for the interval arithmetic. We extend the proof system with news axioms and rules for interval deductions based on the interval arithmetic. This extended domain logic here is called Interval Domain Logic. A goal of this extension is to allow interval deductions which can be used, for example, to solve some Interval Contraint Satisfaction Problems (ICSP). An approach to solve ICSP's is the (Local) Tolerance Propagation [Mac92]. Using the agenda defined by this method we find, via deductions in the proof system, the minimal consistent subinterval which satisfy the equations of the interval constraint system. This method used to find solutions of ICSP's can be implement as an automatic theorem proof.

2 Continuous domains

Several mathematical structures have been widely employs as models for denotational semantics of programming language since the seminal work of Scott and Strachey. One of them is the continuous domains, i.e. posets such that each chain and each consistent set has a supremum, with a least element and a countable base, i.e. basically, a continuous lattice [Sco72] minus, possibly, the top element [Aci96]. The morphisms between continuous domains, called continuous functions, are the monotonic functions w.r.t. the order associated to domains preserving supremum of chains. Continuous domain allows us to develop a theory of interval arithmetic and numerical analysis [Aci91]. This theory has the advantage of being constructive and computational, besides of unifying the theory of programming languages semantics and computational mathematics (numerical analysis). It provides us a logic (Scott logic) to reason about programs in numerical analysis.

Let $\mathbf{D} = \langle D, \leq \rangle$ be a partially ordered set (poset). A set $\Delta \subseteq D$ is called *directed* if each finite subset has an upper bound, or equivalently, $\forall a, b \in \Delta \;\; \exists c \in \Delta$ such that $a \leq c$ and $b \leq c$. A poset $\mathbf{D}$ is *complete* (for short *cpo*) if each directed set Δ has a least upper bound (denoted by $\bigsqcup \Delta$) and has a bottom element. We say that a is *way below* b (denoted by $a \ll b$) if for every directed set Δ such that $b \leq \bigsqcup \Delta$ then $a \leq x$ for some $x \in \Delta$.

We let $\twoheaddownarrow x = \{y \in D : \; y \ll x\}$[1]. A cpo $\mathbf{D}$ is called *continuous* if, for all $x \in D$, the set $\twoheaddownarrow x$ is directed and $x = \bigsqcup \twoheaddownarrow x$. A continuous cpo $\mathbf{D}$ such that to each consistent set (a set has an upper bound in $\mathbf{D}$) has a supremum in D is called *continuous domain.*

The interval data type, denoted here by Υ, will be interpreted by the continuous domain $\mathcal{R} = \langle \mathbb{I}(\mathbb{R}), \sqsubseteq, [-\infty, +\infty] \rangle$ where $\mathbb{I}(\mathbb{R}) = \{[r, s] : r, s \in \mathbb{R} \text{ and } r \leq s\} \cup \{[-\infty, +\infty]\}$ and $[r, s] \sqsubseteq [t, u]$ iff $r \leq t$ and $u \leq s$. Notice that $[a, b] \ll [c, d]$ if, and only if, $a < c$ and $d > b$.

A *topology* on a set X is a collection of subsets of X that is closed under finite intersection and arbitrary union. A set X together with a topology $\mathcal{T}$ on X is a topological space denoted by $(X, \mathcal{T})$. The elements of $\mathcal{T}$ are the open sets of the space. Notice that the emptyset $(= \bigcup \emptyset)$ and X itself $(\bigcap \emptyset)$ are open in any topology over X.

Now let $\langle X, \sqsubseteq \rangle$ be a continuous domain. $\mathcal{O} \subseteq X$ is said to be *Scott open* if whenever $x \in \mathcal{O}$ and $x \sqsubseteq y$ then $y \in \mathcal{O}$ and if $S \subseteq X$ is directed and $Sup\ S \in \mathcal{O}$ then there exists $s \in S$ such that $s \in \mathcal{O}$. The set of Scott open set (the *Scott topology*) of a continuous domain $\mathbf{D}$ is denoted by $\Omega_S(\mathbf{D})$.

3 Continuous information systems

Continuous domains have a more concrete and logic representation as CIS, which can be viewed as prescription or program saying how to build a domain.

Definition 3.1 [*Hoo93*] *A* **continuous information system** *is a triple* $\mathcal{I} = \langle I, Con, \vdash \rangle$, *where* I *is a non empty countable information set,* Con *is a non empty subset of* $\mathcal{P}^{fin}(I)$ *(finite parts of* I*), named consistency predicate, and* $\vdash$ *is a subset of* $Con \times I$*, named entailment relation such that*

I If $X \in Con$ and $Y \subseteq X$ then $Y \in Con$

II If $a \in I$ then, $\{a\} \in Con$

III If $X \vdash a$ then, $X \cup \{a\} \in Con$

IV If $Y \subseteq X$, $X \in Con$ and $Y \vdash a$ then, $X \vdash a$

V If $X \vdash Y$[2] and $Y \vdash a$ then, $X \vdash a$

[1] Analogously, we let $\twoheaduparrow x = \{y \in D : \; x \ll y\}$

[2] The notation $X \vdash Y$ is a shorthand for $\forall a \in Y$, $X \vdash a$.

VI If $X \vdash a$ then $\exists Y \in Con$ such that $X \vdash Y$ and $Y \vdash a$

A CIS informs about elements of a domain which may be identified with the set of information in the system satisfied by these elements.

Definition 3.2 [*Hoo93*] *Let $\mathcal{I} = \langle I, Con, \vdash \rangle$ be a CIS. The set $x \subseteq I$ is an* **element** *of $\mathcal{I}$ if the the following conditions are satisfies:*

1. *If $X \subseteq^{fin} x$*[3] *then, $X \in Con$*
2. *If $X \subseteq x$ and $X \vdash a$ then, $a \in x$.*
3. *If $a \in x$ then $\exists Y \subseteq^{fin} x$ such that $Y \vdash a$.*

The set of elements of a CIS $\mathcal{I} = \langle I, Con, \vdash \rangle$ is denoted by $|\mathcal{I}|$ and the poset $\langle |\mathcal{I}|, \subseteq \rangle$ is called *domain of elements* of the CIS $\mathcal{I}$.

Theorem 3.3 [*Hoo93*] *Let $\mathcal{I} = \langle I, Con, \vdash \rangle$ be a CIS. Then $\langle | \mathcal{I} |, \subseteq \rangle$ is a continuous domain.* ■

There are technical advantages to working with CIS rather than directly with continuous domains. First CIS uses the set theory languages and second the properties of domains can be derived rather than postulated.

We are mainly interested in continuous information system, which are appropriated to representing the real number data type. Each rational interval is interpreted as an information about the real numbers that it properly contains. A CIS for the real numbers is the triple

$$\mathcal{I}_{\mathcal{R}} = \langle \mathbb{I}(\mathbb{Q}), Con_{\mathcal{R}}, \vdash_{\mathcal{R}} \rangle \text{ where}$$

1. $\mathbb{I}(\mathbb{Q}) = \{[a, b] : \ a, b \in \mathbb{Q} \text{ and } a \leq b\} \cup \{[-\infty, +\infty]\}$
2. $X \in Con_{\mathcal{R}}$ iff $X \subseteq^{fin} \mathbb{I}(\mathbb{Q})$ and $max_l(X) < min_r(X)$
3. $X \vdash_{\mathcal{R}} [a, b]$ iff $a < max_l(X)$ e $min_r(X) < b$.

with $max_l(X) = max\{a : \exists b \in \mathbb{Q} \cup \{+\infty\}, \ [a, b] \in X\}$, $min_r(X) = min\{b : \ \exists a \in \mathbb{Q} \cup \{-\infty\}, \ [a, b] \in X\}$. The domain of element of the CIS $\mathcal{I}_{\mathcal{R}}$ is "isomorphic" to the continuous domain $\langle \mathbb{I}(\mathbb{R}), \sqsubseteq \rangle$, where $\mathbb{I}(\mathbb{R}) = \{[r, s] : r, s \in \mathbb{R} \text{ and } r \leq s\} \cup \{[-\infty, +\infty]\}$ and $[r, s] \sqsubseteq [t, u]$ if, and only if, $r \leq t$ and $u \leq s$ [Bed96]. In the follows we will incorporate the interval arithmetics operators in the basic information, i.e. we will consider informations of teh kind $[3, 4] + [2, 3]$ as an information to real number $\pi + e$, for example. Its extension will be useful to obtain a logical system of partial information on the real data type, which allows us to have interessant deduction in the interval theory involving interval arithmetics.

$$\mathcal{I}_{\mathcal{R},A} = \langle I_{\mathcal{R},A}, Con_{\mathcal{R},A}, \vdash_{\mathcal{R},A} \rangle \text{ where}$$

[3] $X \subseteq^{fin} Y$ is an abreviation for "$X \subseteq Y$ and X is finite".

1. $I_{\mathcal{R},A}$ is obtain recursively from the follows formation rules

$$\frac{[a,b] \in \mathbb{I}(\mathbb{Q})}{[a,b] \in I_{\mathcal{R},A}} \quad \frac{I, J \in I_{\mathcal{R},A}}{(I+J) \in I_{\mathcal{R},A}} \quad \frac{I, J \in I_{\mathcal{R},A}}{(I-J) \in I_{\mathcal{R},A}} \quad \frac{I, J \in I_{\mathcal{R},A}}{(I \cdot J) \in I_{\mathcal{R},A}} \quad \frac{I, J \in I_{\mathcal{R},A}}{(I/J) \in I_{\mathcal{R},A}}$$

2. $X \in Con_{\mathcal{R},A}$ iff $eval(X) \in Con_{\mathcal{R}}$
   where $eval : I_{\mathcal{R},A} \longrightarrow \mathbb{I}(\mathbb{Q})$ is a function defined by
   $eval([a,b]) = [a,b]$
   $eval((I+J)) = eval(I) + eval(J)$
   $eval((I-J)) = eval(I) - eval(J)$
   $eval(I \cdot J)) = eval(I) \cdot eval(J)$

$$eval((I/J)) = \begin{cases} eval(I)/eval(J) & \text{if } 0 \notin J \\ \uparrow & \text{otherwise} \end{cases}$$

   The operators $+, -, \cdot$ and $/$, are usual interval arithmetic operators [Moo66].

3. $X \vdash_{\mathcal{R},A} I$ iff $eval(X) \vdash_{\mathcal{R}} eval(I)$.

4 Typed language and formation rules

In this section, we shall introduce a metalanguage for denotational semantics of programs, whose language of the typed expressions has the following syntax:

$$\sigma ::= \mathbf{1} \mid \Upsilon \mid \sigma \times \tau \mid \sigma \to \tau \mid \sigma \oplus \tau \mid \sigma_{\uparrow} \mid \sigma^{H} \mid \sigma^{S} \mid \mathbf{rec}.t.\sigma$$

where t is a variable type, and σ and τ are any types, the type $\mathbf{1}$ consist of a unique element and Υ is the interval type. The product ($\times$), function space ($\to$), collapsed sum ($\oplus$), lifting ($_{\uparrow}$), the Hoare powerdomain (^{H}) and Smyth powerdomain (^{S}) are the usual constructors of domains [Sco82]. To each σ type we associate a CIS, $\mathcal{I}(\sigma) = \langle I_{\sigma}, Con_{\sigma}, \vdash_{\sigma} \rangle$. For example, $\mathcal{I}(\Upsilon) = \mathcal{I}_{\mathcal{R},A}$.

By using the above metalanguage we can provide denotational semantics for a large class of programming languages. Each programming language L is specified by a typed expression σ and each program in it is denoted by an element of $\mathcal{I}(\sigma)$. We are not concerned in how we select a typed expression for a particular programming language.

For each type σ we introduce a propositional language $\mathcal{L}_{\sigma}$, the language of finitely observable informations, whose atomical formulae are I_{σ} and the canonical information $\mathbf{t}$ (true) and $\mathbf{f}$ (false).

1. If $a \in I_{\sigma}$, then $a \in \mathcal{L}_{\sigma}$.

2. If $a, b \in \mathcal{L}_\sigma$, then $a \wedge b \in \mathcal{L}_\sigma$.
3. If $a_1, a_2, \cdots \in \mathcal{L}_\sigma$, then $\bigvee a_i \in \mathcal{L}_\sigma$
4. $\mathbf{t}_\sigma, \mathbf{f}_\sigma \in \mathcal{L}_\sigma$.

Clearly, each language $\mathcal{L}_\sigma$ can be extend with a countable set of variables X, denoted by $\mathcal{L}_\sigma(X)$.

Formation Rule

$$(\mathbf{t},\mathbf{f}) \quad \frac{}{\mathbf{t}:\sigma} \qquad \frac{}{\mathbf{f}:\sigma} \qquad (\wedge) \quad \frac{\varphi, \psi : \sigma}{\varphi \wedge \psi : \sigma} \qquad (\bigvee) \quad \frac{\varphi_1, \varphi_2, \cdots : \sigma}{\bigvee \varphi_i : \sigma}$$

$$(\to) \quad \frac{\psi : \tau}{(\emptyset, \psi) : \sigma \to \tau} \qquad \frac{\varphi : \sigma \quad \psi : \tau}{(\{\varphi\}, \psi) : \sigma \to \tau} \qquad (\times) \quad \frac{\varphi : \sigma \quad \psi : \tau}{(\varphi, \psi) : \sigma \times \tau} \qquad (\text{rec}) \quad \frac{\varphi : \sigma[rec\, X.\sigma/X]}{\varphi : rec\, X.\sigma}$$

$$(()_\perp) \quad \frac{\varphi : \sigma}{(2, \varphi) : \sigma_\perp} \qquad \frac{}{(1,1) : \sigma_\perp} \qquad (\Box) \quad \frac{\varphi : \sigma}{\Box\varphi : \sigma^S} \qquad (\Diamond) \quad \frac{\varphi : \sigma}{\Diamond\varphi : \sigma^H}$$

5 Proof system

In order to give axioms in the program logic, we will introduce a relations for every type σ. By $\varphi \gg \psi$, we mean that the information φ derives the information ψ. So, $\gg$ is an entailment relation or a strong order, also know as "way-below" relation, in an inverse sense.

For notational simplicity, we will eliminate subscripts if no confusion arise and we will use binary disjunction instead of the arbitrary disjunctions. We will introduce for each type the relation $\gg$ on their informations, which indicates the logic derivation.

Logical Axioms and Logical Rules

$$(\mathbf{t}) \quad \frac{}{\varphi \gg \mathbf{t}} \qquad (\mathbf{f}) \quad \frac{}{\mathbf{f} \gg \varphi} \qquad (\text{I-}\gg) \quad \frac{X \vdash b}{\bigwedge X \gg b}$$

$$(\text{I}\geq) \quad \begin{array}{c} [\psi \gg \phi] \\ \vdots \\ \varphi \gg \phi \\ \hline \varphi \geq \psi \end{array} \qquad (\vee - \gg) \quad \frac{\varphi_1 \gg \psi \quad \varphi_2 \gg \psi}{\varphi_1 \vee \varphi_2 \geq \psi} \qquad (\wedge - \gg) \quad \frac{\varphi \gg \psi \quad \varphi \gg \phi}{\varphi \geq \psi \wedge \phi}$$

$$(\text{Trans1}) \quad \frac{\varphi \gg \psi \quad \psi \gg \phi}{\varphi \gg \phi} \qquad (\text{Trans2}) \quad \frac{\varphi \geq \psi \quad \psi \gg \phi}{\varphi \gg \phi} \qquad (\text{Trans3}) \quad \frac{\varphi \gg \psi \quad \psi \geq \phi}{\varphi \gg \phi}$$

$$(\text{Inter}) \quad \frac{\varphi \gg \psi}{\varphi \gg \phi} \quad \text{for some} \quad \phi \gg \psi \qquad (\text{I=}) \quad \frac{\psi \geq \phi \quad \phi \geq \psi}{\psi = \phi}$$

Notice that $\gg$ is defined on the $\vdash$ relation associated to $I(\sigma)$ which is not necessarelly reflexive and we not include the traditional rules for the equality (reflexivity, simmetry and transitivity), because it can be derived from them.

We must introduce several logical axioms and rules in order to construct types. But, as they are not so fundamental to our theory and, moreover they can also be found in [Abr91, BA95], we are omitting them.

In the particular case of the type Υ, the consequence relation on $\mathcal{I}_{\mathcal{R},A}$ when extended using the logical axioms and logical rules, above, will satisfy some interesting properties, as show the follows axioms and rules to the real interval data type.

Axioms and Rules for the Interval Real Type

$$(\mathcal{R}\mathbf{t}) \quad \frac{}{\mathbf{t} = [-\infty, +\infty]} \qquad (\mathcal{R}\mathbf{f}) \quad \frac{b < c}{\mathbf{f} = [a,b] \wedge [c,d]}$$

$$(\mathcal{R}\wedge_1) \quad \frac{[c,b] \gg [a,d]}{[a,b] \wedge [c,d] = [c,b]} \qquad (\mathcal{R}\wedge_2) \quad \frac{[a,d] \gg [c,b]}{[a,b] \wedge [c,d] = [a,d]}$$

$$(\mathcal{R}\vee_1) \quad \frac{[c,b] \gg [a,d]}{[a,b] \vee [c,d] = [a,d]} \qquad (\mathcal{R}\vee_2) \quad \frac{[a,d] \gg [c,b]}{[a,b] \vee [c,d] = [c,b]}$$

$$(\mathcal{R}Mon) \quad \frac{[a,b] \gg [c,d]}{[a,b] \otimes [p,q] \gg [c,d] \otimes [p,q]} \qquad (\mathcal{R}\otimes) \quad \frac{}{[a,b] \otimes [c,d] = [min\ A, max\ A]}$$

$$(\mathcal{R}/_1) \quad \frac{[c,d] \wedge [0,1] = \mathbf{f}}{[a,b]/[c,d] = [minB, maxB]} \qquad (\mathcal{R}/_2) \quad \frac{[c,d] \wedge [-1,0] = \mathbf{f}}{[a,b]/[c,d] = [min\ B, max\ B]},$$

where $A = \{a \otimes c, a \otimes d, b \otimes c, b \otimes d\}$ and $B = \{a/c, a/d, b/c, b/d\}$.

6 Soundness and completeness

The classical Stone representation theorem for Boolean algebras is the prototype for a wide class of "Stone-type duality Theorems" [Joh82, Abr91, AJ94]. The general form of those theorems asserts an equivalence between a category of topological spaces and (the opposite of) a category of lattices and lattices morphisms. The importance of the Stone duality theorems for computer science rests on the fact that it provides the right framework for understanding the relationship between denotational semantics and program logic.

In our case, for a type σ, the Lindenbaum algebra of $\langle \mathcal{L}_\sigma, \sqsubseteq \rangle$, defined as $LA(\sigma) = \langle \mathcal{L}_\sigma / =, \sqsubseteq / = \rangle$[4], is a completely distributive lattice and, therefore, has as Stone dual the Scott topology of a continuous cpo.

Theorem 6.1 (Stone Duality) *$LA(\sigma)$ is the Stone dual of $|I(\sigma)|$, i.e.,*

[4] Here $/=$ denotes module $=$

1. $|\, I(\sigma)\,| \cong \mathbf{Spec} LA(\sigma)$[5]

2. $\Omega_S(|\, I(\sigma)\,|) \cong LA(\sigma)$.■

In other words, this theorem shows that the program logic is equivalent to the usual denotational construction of domains. In this way, a proof of this theorem (part 2) must provide an interpretation function for each (closed) type expression σ. We propose the following

$$[\![\ \ ]\!]_\sigma : \mathcal{L}_\sigma \longrightarrow \Omega_S(|\, I(\sigma)\,|)$$

- $[\![a]\!] = \Uparrow\overline{\{a\}}$, for each $a \in I_\sigma$
- $[\![\mathbf{f}]\!] = \emptyset$
- $[\![\mathbf{t}]\!] = \Uparrow\overline{\emptyset} = |\, S(\sigma)\,|$
- $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$
- $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$

Let $\varphi, \psi \in \mathcal{L}_\sigma$. If $\forall \Uparrow x \subseteq [\![\varphi]\!]$, $\exists \Uparrow y \subseteq [\![\psi]\!]$, such that $y \gg_{|S(\sigma)|} x$, we denote it by $\models \varphi \gg \psi$.

Theorem 6.2 (Soundness and Completeness) *Let σ be a type and $\varphi, \psi \in \mathcal{L}_\sigma$. We have the following*

$$\vdash \varphi \gg \psi \text{ if, and only if, } \models \varphi \gg \psi. \blacksquare$$

7 Application to interval constraint satisfaction problems

The extended continuous domain logic can be useful to solve a kind of interval constraint satisfaction problem. The idea is to decompose the constraint equations up to form only equations with a unique arithmetic operator (sometimes this process requires the introduction of auxilary variables). These new equations are called solution functions [Hyv92]. So, we must consider each tolerance interval constraint and solution functions as hypothesis. These hypothesis are used in the deductions of our proof system to find the minimal interval (the most refined) taken by the variables a such way they continuate satisfying the set of equations.

Consider the conversion from Celsius (C) degrees to Fahrenheit (F) degrees, described by the following equation:

$$F = 1.8 \cdot C + 32 \tag{1}$$

If C (input) is known, then F (output) can be computed by combining two local computations $X = 1.8 \cdot C$ and $F = X + 32$. By applying the inverse local function

[5] Given a completely distributive lattice, $\mathbf{Spec}A$, the spectrum of the lattice A, is the space of all prime elements $p \neq \mathbf{t}$ of A endowed with the hull-kernel topology [GHK+80].

$X = F - 32$ and $C = X/1.8$ we can compute C from F. An example of ICSP is obtains the minimal interval values of C with F satisfying the above equation and such that refines the initial interval temperature measurements $C = [1, 5]$ and $F = [27, 35]$.

In the (local) tolerance propagation method [Hyv92] this is solved via a procedure which starts with the values for the variables $C := [1, 5]$, $X := [-\infty, +\infty]$ and $F := [27, 35]$. These values correspond to the tolerance constraints of the variable values. Product of decompositions of the original equation will give two equations constraints and four solution functions corresponding to these constraints.

constraints	solution functions
$(C1)\ C \cdot 1.8 = X$	$(E1)\ \ X = 1.8 \cdot C$, $(E2)\ \ C = X/1.8$
$(C2)\ X + 32 = F$	$(E3)\ \ F = X + 32$, $(E4)\ \ X = F - 32$

The initial solution functions agenda of this network is, for example, $X = 1.8 \cdot C$, $X = F - 32$, $C = X/1.8$ and $F = X + 32$. In this cases, the values 1.8 and 32 are abreviations of the degenerate intervals $[1.8, 1.8]$ and $[32, 32]$. The final values for the variables after the tolerance propagation algorithm in [Hyv92] with this agenda is applied are: $C := [1, 1.\overline{6}]$, $X := [1.8, 3]$ and $F := [33.8, 35]$.

In order to obtain these values, in our proof system, we extend the language to $\mathcal{L}_\sigma(V)$, where $V = \{F, C, X\}$. For that we have to introduce as hypothesis the interval constraint tolerance $C \gg [1, 5]$, $X \gg \mathbf{t}$, $F \gg [27, 35]$.

(D1)Deduction of the auxiliar variable X value.

$$\cfrac{\cfrac{\cfrac{}{X\gg 1.8\cdot C}(E1)\quad \cfrac{\cfrac{}{C\gg[1,5]}(Hyp)}{1.8\cdot C\gg[1.8,9]}(\mathcal{R}\otimes)}{X\gg[1.8,9]}(trans1)\qquad \cfrac{\cfrac{}{X\gg F-32}(E4)\quad \cfrac{\cfrac{}{F\gg[27,35]}(Hyp)}{F-32\gg[-5,3]}(\mathcal{R}\otimes)}{X\gg[-5,3]}(trans1)}{X\gg[1.8,9]\wedge[-5,3]}(\wedge-\gg)$$

Our deduction continuates in the following proof tree

$$\cfrac{\cfrac{}{X\gg[1.8,9]\wedge[-5,3]}D1\qquad \cfrac{}{[1.8,9]\wedge[-5,3]=[1.8,3]}(\mathcal{R}\wedge)}{X\gg[1.8,3]}(trans3)$$

(D2) Deduction for obtaining the value of the variable C

$$\cfrac{\cfrac{}{C\gg X/1.8}(E2)\qquad \cfrac{\cfrac{}{X\gg[1.8,3]}(D1)}{X/1.8\gg[1,1.\overline{6}]}(\mathcal{R}\otimes)}{C\gg[1,1.\overline{6}]}(trans1)$$

(D3) Deduction of the variable F value.

$$\cfrac{\cfrac{}{F\gg X+32}(E3)\qquad \cfrac{\cfrac{}{X\gg[1.8,3]}(D1)}{X+32\gg[33.8,35]}(\mathcal{R}\otimes)}{F\gg[33.8,35]}(trans1)$$

In our next example, the goal is to solve, in our proof system, a project problem. The task is to alocate the work time of n researchers to m projects. For example, consider three researchers, a, b, and c, with capacity of work hours A, B and C, respectively. These researchers must be alocated to the projects p_1, p_2 and p_3 which require resources of P_1, P_2 and P_3 hours, respectively. Let ap be a variable denoting the work time of the researcher a spended in the project p. The equations of this $ICSP$ are as follows [Hyv92]:

- E1) $A + B + C = P_1 + P_2 + P_3$
- E2) $T = P1 + P2 + P3$
- E3) $P_1 = ap_1 + bp_1 + cp_1$
- E4) $P_2 = ap_2 + bp_2 + cp_2$
- E5) $P_3 = ap_3 + bp_3 + cp_3$
- E6) $A = ap_1 + ap_2 + ap_3$
- E7) $B = bp_1 + bp_2 + bp_3$
- E8) $C = cp_1 + cp_2 + cp_3$

A possible set of tolerance constraints to the variables are: $ap_1 = [120, 160]$, $bp_1 = [0, 160]$ and $cp_1 = [0, 160]$. The remains are assumed to have no constraints. Just is presented a proof tree to determinate the hours consumed by the project P_1, satisfying these constraints.

$$\dfrac{\dfrac{}{P_1{=}ap_1{+}bp_1{+}cp_1}(E3) \quad \dfrac{\dfrac{\dfrac{}{ap_1{\gg}[120,160]}(Hyp) \quad \dfrac{}{bp_1{\gg}[0,160]}(Hyp)}{ap_1{+}bp_1{\gg}[120,320]}(\mathcal{R}\otimes) \quad \dfrac{}{cp_1{\gg}[0,160]}(Hyp)}{ap_1{+}ap_2{+}ap_3{\gg}[120,480]}(\mathcal{R}\otimes)}{P_1{\gg}[120,480]}(trans2)$$

If the tolerance constraints are inconsistent, we can only deduce on the proof system the interval $[-\infty, +\infty]$. For example, if the tolerance constraints are $T = [0, 160]$, $P_1 = [120, 160]$, $P_2 = [0, 160]$ and $P_3 = [100, 160]$, then we have the following deduction:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{T{=}P_1{+}P_2{+}P_3}(E2) \quad \dfrac{\dfrac{\dfrac{}{P_1{\gg}[120,160]}(Hyp) \quad \dfrac{}{P_2{\gg}[0,160]}(Hyp)}{P_1{+}P_2{\gg}[120,320]}(\mathcal{R}\otimes) \quad \dfrac{}{P_3{\gg}[100,160]}(Hyp)}{P_1{+}P_2{+}P_3{\gg}[220,480]}(\mathcal{R}\otimes)}{T{\gg}[220,480]}(Trans2) \quad \dfrac{}{T{\gg}[0,160]}(Hyp)}{T{\gg}[220,480]\wedge[0,160]}(\wedge - \gg)}{T{\gg}\mathbf{f}}(\mathcal{R}\mathbf{f})$$

8 Conclusion

In this paper we use the continuous domain logic introduced by [BA95], in order to develop a program logic for interval analysis. This program logic agree with the denotational semantics in the sense that the denotation of any program coincides with the set of true assertions of it. So, we provide a behaviour study of interval programming languages. On the other hand, we can use this logic to perform interval deductions. In this sense we incorporate in the domain logic the interval data type Υ as primitive. Since the interval arithmetic is fundamental to the interval analysis, we have included it in the proper definition of their continuous information system and associate it to type Υ. Clearly, both continuous information systems given by the real interval domain are isomorphics. We believe that the extended domain logic presented here, denominated interval domain logic, allows us to solve a large class of problems in interval analysis, such as linear systems, ICSP, etc.

In particular, in our logic we gave two examples of deductions, which allow us to solve two ICSPs. We believe that this kind of deduction can be applied to all ICPS which can be solved by this method, because the estrategy used to find the proof tree is similar to local tolerance propagation techniques. We are not saying that we have provided a formal method to solve any ICSP. In fact we have just solved simple ICSP's. But, the way how we have found solutions in interval constraint systems is general (at most to local propagation techniques) and computable. So, we hope our approach can be implemented in automatic theorem proof on interval domain logic to solve a large class of ICSP.

References

[Abr91] Samson Abramsky. *Domain Theory in Logic Form.* In Annals of Pure and Applied Logic, 51:1-77, 1991.

[Aci91] Benedito Melo Acióly. *Computational Foundation of Interval Mathematics* (in portuguese). PhD thesis, CPGCC da UFRGS, Porto Alegre, 1991.

[Aci96] Benedito Melo Acióly. *The Scott Interval Analysis.* In II workshop of Arithmetic, Interval and Symbolic Computations - WAI96. Recife, august of 1996. pages 4-6.

[AJ94] Samson Abramsky and Achim Jung. Domain Theory. In *Handbook of Logic in Computer Science*, Vol. 3, Oxford university Press., 1994.

[BA95] Benjamín R. Callejas Bedregal and Benedito Melo Acióly. *Logic of Plotkin Continuous Domain.* In LNCS 911, pages 195-206, Springer-Verlag, 1995.

[Bed96] Benjamín R. Callejas Bedregal. *Continuous Information Systems: A Computational and Logical Approach to Interval Mathematics* (in portuguese). PhD thesis, UFPE-Depto. de Informática, Recife, 1996.

[GHK+80] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove and D. Scott. *A Compemdium of Continuous Lattices.* Springer-Verlag, Berlim, 1980.

[Gun85] Carl A. Gunter. *Comparing categories of domains.* In LNCS 239, pages 101-121. Springer-Verlag, 1985.

[Hoo93] R. Hoofman. *Continuous Information Systems.* Information and Computation, 105(1):42-71, 1993.

[Hyv92] Eero Hyvönen. *Constraint Reasoning Based on Interval Arithemetic: The tolerance propagation approach.* In Artificial Intelligence, 58(1-3):71-112, North-Holland, 1992.

[Joh82] Peter T. Johnstone. *Stone Space.* Vol. 3 of Cambridge Studies in Dvances Mathematics. Cambridge University Press., Cambridge-UK, 1982.

[Mac92] Alan K. Mackworth. *The Logic of Constraint Satisfaction.* In Artificial Intelligence, 58(1-3):3-20, North-Holland, 1992.

[Moo66] Ramon E. Moore. *Interval Analysis.* Englewoods Cliffs: Prentice Hall, 1966.

[Sco72] Dana Scott. *Continuous Lattices.* Lectures Notes in Mathematics, 274, pages 97-136, Springer-Verlag, 1972.

[Sco82] Dana Scott. *Domain for denotational semantics.* In LNCS, 140, Springer-Verlag, 1982.

[Vic89] S.J. Vickers. *Topology via Logic.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press., Cambridge-UK, 1989.