# Un Enfoque Genérico y Cooperativo para la Resolución de Restricciones de Intervalo

# (Resumen en español de la Tesis Doctoral)

Por

**Antonio J. Fernández Leiva**

Resumen en español de la tesis de doctorado
remitida en inglés
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, España

Director: Dr. Patricia M. Hill
School of Computing
University of Leeds, England

Tutor: Dr. José M. Troya
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, España

25 de Enero de 2002

# Siglas referenciadas

Hemos respetado las siglas tal y como suelen usarse en la lengua inglesa.

| *Siglas* | *Significado* |
|---|---|
| AI | Artificial Intelligence |
| ATOAM | yet Another matching Tree Oriented Abstract Machine |
| CCL | Concurrent Constraint Logic |
| CCP | Concurrent Constraint Programming |
| CHR | Constraint Handling Rule |
| CLP | Constraint Logic Programming |
| CLP(B) | Constraint Logic Programming on the Boolean domain |
| CLP(FD) | Constraint Logic Programming on the finite domain |
| CLP($\Re$) | Constraint Logic Programming on the real domain |
| CP | Constraint Programming |
| CSP | Constraint Satisfaction Problem |
| FD | Finite Domain |
| LP | Logic Programming |
| OR | Operational Research |
| RC | Reified Constraint |
| SRQ | Self Referential Quiz |
| WAM | Warren Abstract Machine |
| 2D | 2-Dimensional |

# Sinopsis

Esta tesis propone un esquema genérico y cooperativo para CLP(Interval($\mathcal{X}$)) donde $\mathcal{X}$ es cualquier dominio de computación con estructura de retículo. El esquema, que está basado en la teoría de retículos, es un enfoque general para la satisfacción y optimización de restricciones de intervalo así como para la cooperación de resolutores de intervalo definidos sobre dominios de computación con estructura de retículos, independientemente de la cardinalidad de éstos. Nuestra propuesta asegura un enfoque transparente sobre el cual las restricciones, los dominios de computación y los mecanismos de propagación y cooperación, definidos entre las variables restringidas, pueden ser fácilmente especificados a nivel del usuario. La parte principal de la tesis presenta una especificación formal de este esquema.

Los principales resultados conseguidos en esta tesis son los siguientes:

- Una comparativa global de la eficiencia y algunos aspectos de la expresividad de ocho sistemas de restricciones. Esta comparativa, realizada sobre el dominio finito y el dominio Booleano, muestra diferencias principales entre los sistemas de restricciones existentes.

- Para formalizar el marco de satisfacción de restricciones para CLP(Interval($\mathcal{X}$)) hemos descrito el proceso global de resolución de restricciones de intervalo sobre cualquier retículo, separando claramente los procesos de propagación y división (ramificación) de intervalos. Una de las ventajas de nuestra propuesta es que la monotonía de las restricciones está implícitamente definida en la teoría. Además, declaramos un conjunto de propiedades interesantes que, bajo ciertas condiciones, son satisfechas por cualquier instancia del esquema genérico. Más aún, mostramos que muchos sistemas de restricciones actualmente existentes satisfacen estas condiciones y, además, proporcionamos indicaciones sobre cómo extender el sistema mediante la especificación de otras instancias interesantes y novedosas.

- Nuestro esquema para CLP(Interval($\mathcal{X}$)) permite la cooperación de resolutores de manera que la información puede fluir entre diferentes dominios de computación. Además, es posible combinar distintas instancias del esquema: por ejemplo, instancias bien conocidas tales como CLP(Interval($\Re$)), CLP(Interval(*Integer*)), CLP(Interval(*Set*)), CLP(Interval(*Bool*)), y otras novedosas que son el resultado de la generación de nuevos dominios de computación definidos por el usuario, o incluso que surgen de la combinación de dominios ya existentes como puede ser

CLP(Interval($\mathcal{X}_1 \times \ldots \times \mathcal{X}_n$)). Por lo tanto, $\mathcal{X}$ puede ser instanciado a cualquier conjunto de dominios de computación con estructura de retículo de forma que su correspondiente instancia CLP(Interval($\mathcal{X}$)) permite una amplia flexibilidad en la definición de dominios en $\mathcal{X}$ (probablemente definidos por el usuario) y en la interacción entre estos dominios.

- Mediante la implementación de un prototipo, demostramos que un único sistema, que esté basado en nuestro esquema para CLP(Interval($\mathcal{X}$)), puede proporcionar soporte para la satisfacción y la optimización de restricciones así como para la cooperación de resolutores sobre un conjunto conteniendo múltiples dominios de computación. Además, el sistema sigue un novedoso enfoque transparente sujeto a una doble perspectiva ya que el usuario puede definir no sólo nuevas restricciones y su mecanismo de propagación, sino también nuevos dominios sobre los cuales nuevas restricciones pueden ser resueltas así como el mecanismo de cooperación entre todos los dominios de computación (ya sean definidos por el usuario o predefinidos por el sistema).

  En nuestra opinión, esta tesis apunta nuevas y potenciales direcciones de investigación dentro de la comunidad de las restricciones de intervalo.

Para alcanzar los resultados expuestos, hemos seguido los siguientes pasos (1) la elección de un enfoque adecuado sobre el cual construir los fundamentos teóricos de nuestro esquema genérico; (2) la construcción de un marco teórico genérico (que llamaremos el *marco básico*) para la propagación de restricciones de intervalo sobre cualquier retículo; (3) la integración, en el marco básico, de una técnica novedosa que facilita la cooperación de resolutores y que surge de la definición, sobre múltiples dominios, de operadores de restricciones y (4) la extensión del marco resultante para la resolución y optimización completa de las restricciones de intervalo.

Finalmente presentamos *clp*($\mathcal{L}$), un lenguaje de programación lógica de restricciones de intervalo que posibilita la resolución de restricciones sobre cualquier conjunto de retículos y que está implementado a partir de las ideas formalizadas en el marco teórico. Describimos una primera implementación de este lenguaje y desarrollamos algunos ejemplos de cómo usarla. Este prototipo demuestra que nuestro esquema para CLP(Interval($\mathcal{X}$)) puede ser implementado en un sistema único que, como consecuencia, proporciona, bajo un enfoque transparente sobre dominios y restricciones, cooperación de resolutores así como satisfacción y optimización completa de restricciones sobre diferentes dominios de computación.

## Publicaciones

Las motivos que han dado lugar a la tesis, y que son descritos en la Parte I, fueron originalmente presentados en (Fernández, 1997).

La Parte II ha sido publicada, casi íntegramente, en la revista Internacional *Constraints* (Fernández y Hill, 2000a). Con anterioridad, las secciones sobre puzzles auto-referenciales fueron presentadas en (Fernández y Hill, 1997a) y (Fernández y

Hill, 1997b). Un versión mas corta de la comparativa en eficiencia fue publicada en (Fernández y Hill, 1998c).

Con respecto a la Parte III, una versión preliminar del Capítulo 3 se publicó en (Fernández y Hill, 1999b). Otras versiones previas, y menos maduras, de este capítulo fueron presentadas en (Fernández y Hill, 1998b; Fernández y Hill, 1998a) y (Fernández y Hill, 1999a). Los capítulos 4 y 5 se expusieron en (Fernández y Hill, 2000b) y (Fernández y Hill, 2001a) respectivamente. Un versión de revista que integra los capítulos 3 y 4 se encuentra actualmente en fase de revisión (Fernández y Hill, 2001b).

La parte IV se encuentra disponible como manual de usuario (Fernández, 2000).

## Notas del autor

Todas las demostraciones de teoremas, lemas y proposiciones han sido omitidas en este documento, puesto que el objetivo del mismo es aportar un resumen (extendido) de la tesis completa. Si algún lector está interesado en las demostraciones, éstas pueden encontrarse en la versión original y completa de esta tesis que se ha escrito en inglés y que estará, próximamente, disponible en la página Web del autor, la cual se encuentra en la siguiente dirección

`http://www.lcc.uma.es/~afdez/`

Con respecto a la versión inglesa, las siguientes equivalencias deben ser tenidas en cuenta, sobre todo al consultar las demostraciones:

| *Español* | *Inglés* | *Significado* |
|---|---|---|
| *propaga* | *solve* | Procedimiento de propagación |
| *resuelve* | *branch* | Procedimiento de ramificación |
| *selecciona* | *choose* | Función de selección |
| *divide* | *split* | Función de partición |
| *coloca* | *push* | Operación de pila |
| *cima* | *top* | Operación de pila |
| *Pilas* | *Stack* | Conjunto de pilas |

# Índice general

**Siglas** III

**Resumen** V

**I Introducción** 1

**1. Introducción y Motivación** 3

**II Marco Comparativo** 11

**2. Una Comparación de Sistemas Transparentes** 13
2.1. Introducción y Motivación . . . . . 13
2.2. Evaluando la Expresividad . . . . . 15
2.2.1. Restricciones Reificadas . . . . . 15
2.2.2. Meta-restricciones . . . . . 16
2.2.3. Valoración de los Resultados . . . . . 16
2.2.4. CHR: una Mención Especial . . . . . 17
2.3. Evaluando el Rendimiento . . . . . 17
2.4. Conclusiones . . . . . 18
2.5. Contribuciones . . . . . 19

**III Marco Teórico** 21

**3. Propagación de Restricciones de Intervalo sobre Retículos de Intervalo** 23
3.1. Motivaciones . . . . . 23
3.2. Conceptos Preliminares y Notación . . . . . 25
3.3. Los Dominios de Computación y el de Intervalo . . . . . 27
3.3.1. Dominios de Computación Acotados . . . . . 28
3.3.2. Operadores de Restricción . . . . . 30
3.3.3. Indexicals . . . . . 31

3.3.4. El Dominio de Intervalos . . . . . 32
3.4. Los Dominios de las Restricciones . . . . . 34
3.4.1. Restricciones de Intervalo . . . . . 34
3.4.2. Reducción de Restricciones . . . . . 37
3.4.3. Propagación de Restricciones . . . . . 38
3.4.4. Equivalencia en el Dominio Discreto . . . . . 40
3.4.5. Una Solución para un Almacén de Restricciones . . . . . 41
3.4.6. Monotonía de las Restricciones . . . . . 41
3.5. Semántica Operacional . . . . . 43
3.5.1. Esquema Operacional para la Propagación de Restricciones . . . 43
3.5.2. Terminación . . . . . 44
3.6. Instancias . . . . . 47
3.6.1. Dominios Clásicos . . . . . 47
3.6.2. Otros ejemplos . . . . . 48
3.6.3. Combinaciones de Dominios . . . . . 48
3.7. Contribuciones . . . . . 49

**4. Cooperación de Resolutores de Intervalo 51**
4.1. Introducción y Motivaciones . . . . . 51
4.2. Restricciones de Alto Nivel . . . . . 52
4.3. Ejemplos . . . . . 54
4.3.1. Combinaciones Lineales . . . . . 54
4.3.2. Incluso más Expresividad! . . . . . 55
4.4. Conclusiones . . . . . 56
4.5. Contribuciones . . . . . 57

**5. Ramificación de Restricciones de Intervalo 59**
5.1. Introducción . . . . . 59
5.2. Algunos Conceptos Importantes . . . . . 60
5.3. El Proceso de Ramificación . . . . . 62
5.3.1. La Función de Precisión como una Regla de Normalización . . . 64
5.4. Ramificación en la Resolución de las Restricciones de Intervalo . . . . . 64
5.5. Problemas de Optimización (COPs) . . . . . 66
5.5.1. Formas Diversas de Resolver las Instancias . . . . . 68
5.6. Conclusiones . . . . . 69
5.7. Contribuciones . . . . . 70

**IV Marco Práctico 73**

**6. Un lenguaje de CLP Genérico, Colaborativo y de 2D 75**
6.1. Motivación . . . . . 75
6.2. Conclusiones . . . . . 76
6.3. Contribuciones . . . . . 76

**7. Conclusiones** **79**
7.1. Resumen de los Resultados . . . . . 79
7.2. Resumen de las Principales Contribuciones . . . . . 81
7.3. Trabajo Futuro . . . . . 81

**Bibliografía** **83**

# Índice de figuras

1.1. Razonamiento específico sobre dominios distintos . . . . . 4
1.2. El enfoque de un resolutor genérico . . . . . 6
1.3. Un enfoque transparente bidimensional: sobre las restricciones y sobre los dominios de computación . . . . . 9

3.1. Relación entre dos rangos $\overline{s_1}, t_1$ y $\overline{s_2}, t_2$ de $R_L^s$ para algún $L \in \mathcal{L}$ . . . . . 33
3.2. Estructura del dominio de intervalos simple $R_L^s$ donde $a, b, c \in L$ y $a \prec_L c \prec_L b$ . . . . . 35
3.3. *propaga*/2: un esquema genérico para la propagación de las restricciones de intervalo . . . . . 43

4.1. Combinaciones lineales de la suma de dominios . . . . . 54

5.1. $resuelve_\alpha$/3: un esquema genérico de ramificación para la resolución de las restricciones de intervalo . . . . . 66

# Índice de cuadros

5.1. El tipo de CSP depende de la instanciación de los parámetros . . . . . 68

# Parte I

# Introducción

# Capítulo 1

# Introducción y Motivación

*It isn't that they can't see the solution.*
*It is that they can't see the problem.*

The Scandal of Father Brown (1935)
G.K. Chesterton, 1874-1936

La programación lógica con restricciones (CLP) (Jaffar y Maher, 1994) nace a partir de una necesidad de resolver los problemas que la programación lógica (LP) no puede resolver. El éxito de CLP radica en la combinación de la declaratividad de LP con la eficiencia mostrada por el paradigma de la programación con restricciones (CP). Sin embargo, aunque ya se ha realizado una considerable investigación en este área, CLP es un campo de investigación relativamente nuevo y en el que los sistemas presentan todavía evidentes limitaciones.

En este capítulo discutimos estas limitaciones y proponemos posibles soluciones para cada una de ellas. Proporcionar un marco teórico adecuado para estas soluciones en un entorno único es el principal objetivo de esta tesis que, en definitiva, consiste en encontrar un sistema genérico, cooperativo, transparente y eficiente para CLP.

**Un esquema genérico para CLP.** El componente esencial del esquema de CLP es que puede ser parametrizado por un dominio de computación de tal manera que diferentes tipos de dominio determinan diferentes instancias del esquema; por ejemplo, tenemos CLP(FD) (CLP sobre intervalos finitos de enteros), CLP($\Re$) (CLP sobre el dominio de los reales), CLP(Set) (CLP sobre conjuntos finitos de elementos) y CLP(Bool) (CLP sobre el dominio Booleano). Los sistemas para CLP han sido aplicados a muchos y diferentes dominios, y cada instancia de CLP puede ser identificada por el mecanismo de resolución de restricciones utilizado y por la estructura algebraica del dominio. El poder expresivo y la eficiencia de los sistemas lógicos con restricciones se reduce debido a la diferente naturaleza de las estructuras (i.e., los objetos -valores en el dominio

de computación- y las operaciones definidas sobre ellos) en las cuales las restricciones pueden ser expresadas. Esto quiere decir que las restricciones tienen ser formuladas sobre un dominio específico y resueltas por el método de razonamiento y el resolutor asociado a ese dominio (véase Figura 1.1). En particular en los actuales sistemas de CLP, la cardinalidad del dominio determina la forma del procedimiento de resolución y, como consecuencia, hay enfoques diferentes para los dominios finitos y los infinitos.

Figura 1.1: Razonamiento específico sobre dominios distintos

La ventaja de un resolutor específico para un dominio es que puede ser optimizado teniendo en cuenta las características particulares de la estructura algebraica del dominio, lo cual se traduce en una mejora del rendimiento del resolutor. Sin embargo, los resolutores específicos tienen también desventajas obvias:

- Primero un resolutor específico no es capaz de resolver restricciones definidas

sobre otros dominios de computación y, por tanto, su contexto de aplicación se restringe al dominio para el cual fue diseñado. Esta es una limitación muy importante ya que, en la práctica, los problemas no son específicos a un dominio de computación. Como consecuencia la formulación de un problema tiene que ser artificialmente adaptada para que pueda ser resuelta por un resolutor específico, perdiendo pues parte de la declaratividad de la solución.

- Segundo, a pesar del hecho de que la semántica (declarativa y operacional) de un resolutor, podría derivarse a partir del esquema básico de CLP, hay todavía un buen número de propiedades (e.g., terminación, corrección y completitud) que tienen que ser probadas (y demostradas) para cada resolutor de forma independiente, es decir, el procedimiento operational implementado específicamente para el resolutor tiene que ser explicado de una manera particular y no puede ser deducido a partir del comportamiento de otros resolutores. Más aún, no siempre está claro cuáles son las principales propiedades de aspectos claves del resolutor (e.g., las heurísticas usadas) que son responsables de propiedades específicas del resolutor.

Estas desventajas desaparecen cuando se considera un enfoque genérico. Por esta razón, en los últimos años han aparecido un cierto número de propuestas con el fin de encontrar principios generales bajo los cuales las principales propiedades de los algoritmos de resolución de restricciones pueden ser explicadas y el comportamiento operacional de las diferentes instancias del esquema puede ser deducido. Además, un resolutor genérico permite que un problema pueda ser formulado en términos de restricciones definidas sobre varios dominios de computación, lo que significa que el código final será más parecido a la formulación original del problema.

A partir de esta discusión se deduce que lo que se necesita es un sistema basado en un resolutor que ejecute resolución de restricciones independientemente de la naturaleza y cardinalidad del dominio de computación (véase la Figura 1.2). Naturalmente este sistema tiene que servir como un marco unificado que incluya los dominios mas tradicionales usados en CLP, es decir, los enteros, los Booleanos, los conjuntos y los reales.

**Un enfoque transparente más amplio.** En los actuales sistemas muchas de las restricciones están controladas directamente por el sistema. Estas *restricciones opacas* proporcionan herramientas muy eficientes que pueden usarse en aplicaciones comunes. Sin embargo existen dos problemas prácticos con este enfoque.

- El proceso de resolución está normalmente predefinido e implementado a bajo nivel, lo cual hace difícil, sino imposible, construir nuevos resolutores sobre nuevos dominios.

- Un resolutor opaco carece de la suficiente flexibilidad para ser usado en aplicaciones no convencionales ya que el problema tiene que ser codificado mediante restricciones predefinidas que son proporcionadas por el sistema. En definitiva,

Figura 1.2: El enfoque de un resolutor genérico

el grado de libertad que el usuario tiene para la codificación de un problema está limitado por los constructores predefinidos y específicos del sistema.

Para solventar esta carencia de flexibilidad, algunos sistemas proporcionan hoy en día restricciones transparentes las cuales pueden ser definidas por el usuario, aunque en la mayoría de los casos, la definición de las estas restricciones se limita todavía a los dominios predefinidos por el sistema, tales como los enteros. La ventaja de un sistema transparente es que el usuario puede definir las restricciones que necesite con el objetivo de conseguir la mejor formulación de un problema. Además, en este tipo de sistemas el usuario normalmente también puede especificar el mecanismo de propagación de las restricciones lo que suele traducirse en programas más eficientes.

Sin embargo, en los actuales sistemas transparentes los resolutores están limitados a los dominios predefinidos lo cual limita la flexibilidad de este enfoque ya que (como ya ha sido apuntado), en la práctica, los problemas son heterogéneos y, a menudo, su formulación natural usa dominios de computación que son diferentes a los dominios predefinidos por el sistema. Como consecuencia, estos sistemas siguen careciendo de la expresividad necesaria y, por tanto, el código final de la solución no ofrece muy a menudo una gran similitud con la declaración inicial del problema.

Una solución a este problema consiste en ampliar el típico enfoque transparente sobre las restricciones a los dominios de computación. Con este enfoque más amplio, el usuario podría definir nuevos dominios, nuevas restricciones sobre estos dominios y el esquema de propagación de los valores de los dominios. Nosotros llamamos a este nuevo enfoque el *enfoque transparente bidimensional* pues, como se muestra en la Figura 1.3, crece en dos direcciones diferentes (verticalmente, mediante la definición de planos que representan los nuevos dominios definidos por el usuario y, horizontalmente, sobre cada plano, lo cual simboliza la generación de nuevas restricciones definidas por el usuario sobre cada dominio de computación).

Para resolver todos los problemas esbozados hasta ahora, combinamos los requisitos del enfoque transparente bidimensional con ésos declarados para el resolutor genérico, y deducimos que lo que se necesita es un sistema de programación lógica con restricciones que sea genérico y se implemente siguiendo el enfoque transparente bidimensional.

**Un esquema colaborativo.** Como ya hemos comentado, la mayoría de los sistemas lógicos con restricciones sólo ofrecen resolución sobre dominios específicos, incluso en el caso de que el sistema siga un enfoque transparente. Esto implica que las restricciones, incluso si son definidas por el usuario, deben ser formuladas exclusivamente sobre elementos pertenecientes al dominio de computación subyacente. For ejemplo, una restricción aritmética tal como $x + y = z$ debe ser definida de forma que todos los elementos $x, y$ y $z$ pertenezcan al mismo dominio (e.g., entero o real). Sin embargo, como ya se apuntó anteriormente, muchos problemas son expresados mas naturalmente usando restricciones heterogéneas definidas sobre más de un dominio. Incluso existen restricciones definidas sobre multiples dominios que requieren la cooperación de dominios distintos a través del envío y la recepción de información de un dominio a otro completamente diferente (e.g., $w = x > y$). Esto significa que, aún en el caso de sistemas sujetos al enfoque bidimensional, la formulación de los problemas reales tiene, de nuevo, que ser artificialmente adaptada a un dominio único (que es uno de los soportados por el sistema, bien sea predefinido o definido por el usuario).

Una solución a este problema está en el concepto de *colaboración de resolutores* que combina la cooperación con la combinación de resolutores. Por una parte, la cooperación de resolutores permite que las restricciones sean propagadas desde un dominio de computación a otro distinto permitiendo que la información fluya entre diferentes dominios de computación. Por otra parte, la combinación de resolutores posibilita la mezcla de resolutores y/o la generación de nuevos resolutores sobre dominios combinados. En general, la colaboración de resolutores ayuda a solventar dos problemas: la

carencia de declaratividad de las soluciones y el pobre rendimiento de los sistemas. Sin embargo, la mayoría del trabajo realizado sobre este campo está enfocado sobre dominios predefinidos y no es flexible ni adaptable a otros dominios. Una solución a este problema es el enfoque bidimensional discutido anteriormente.

De nuevo, enlazando todos los requisitos ya discutidos, deducimos que lo que realmente se necesita es un sistema genérico que permita la cooperación y combinación de resolutores definidos sobre distintos dominios de computación (posiblemente definidos por el usuario) sobre los cuales el esquema de propagación pueda ser definido en el nivel del usuario.

**Un esquema eficiente.** La mayoría de los sistemas actuales de CLP no satisfacen todos los requerimientos discutidos hasta ahora. La principal excepción es el lenguaje CHR (Frühwirth, 1998), el cual permite a los usuarios definir nuevos dominios y nuevas restricciones así como el mecanismo de interacción entre los mismos. Desafortunadamente, este resolutor es ineficiente (Fernández y Hill, 2000a) y su rendimiento es pobre en comparación con la mayoría de los sistemas de restricciones específicos a algún dominio.

La principal crítica a los enfoques genéricos es que la eficiencia es una de las razones principales para escoger la CLP y los resolutotes genéricos raramente alcanzan la velocidad de los resolutores específicos. Sin embargo, un resolutor genérico también aporta importantes ventajas. Una de ellas es que cualquier mejora global en la eficiencia, probablemente, dará lugar a un mejor rendimiento de cada una de las instancias del sistema genérico. También, desde el punto de vista teórico, un enfoque genérico nos permite investigar los principios generales para la resolución de las restricciones y, por lo tanto, estudiar propiedades generales que son satisfechas por todas y cada una de sus instancias.

A pesar de que la mayoría de los resolutores se basan en extensiones de las técnicas de consistencia descritas en (Mackworth, 1977), el razonamiento resolutivo difiere para cada dominio de computación. En general hay dos formas básicas de razonamiento: el de dominio y el de intervalo. El razonamiento de intervalo (particularizado a los límites del dominio) es más eficiente que el razonamiento de dominio (que razona sobre todos los valores del dominio). Como consecuencia, si requerimos eficiencia, el razonamiento de intervalo parece ser más adecuado para un enfoque genérico.

Si añadimos el razonamiento de intervalo a los requerimientos previos para construir nuestro sistema ideal para CLP, vemos que lo que se necesita es un sistema lógico con restricciones que sea genérico, colaborativo, transparente y basado en razonamiento 'intervalar'. En definitiva, esta tesis describe la formalización de un sistema satisfaciendo todos estos requerimientos.

Figura 1.3: Un enfoque transparente bidimensional: sobre las restricciones y sobre los dominios de computación

# Parte II

# Marco Comparativo

# Capítulo 2

# Una Comparación de Sistemas Transparentes

*Your true value depends entirely on what you are compared with.*

Bob Wells

## 2.1. Introducción y Motivación

La evidencia del éxito del paradigma de CLP (Csontó y Paralič, 1997; Jaffar y Lassez, 1987) se encuentra en el número creciente de sistemas para CLP que están, hoy día, siendo usados sobre aplicaciones reales (PAPPACT'98, 1998). En general hay dos razones principales para este éxito: la primera es que CLP extiende el paradigma de la programación lógica permitiendo soluciones más declarativas y legibles y, la segunda, es que proporciona propagación de restricciones sobre dominios específicos, con lo cual se generan implementaciones eficientes de los procedimientos operacionales. Sin embargo, CLP difiere significativamente tanto en la forma en la cual las soluciones pueden ser expresadas como en la eficiencia de la solución (es decir, la eficiencia mostrada a la hora de ejecutar la solución). Es importante que ambos factores sean tenidos en cuenta a la hora de escoger el mejor sistema de CLP para la resolución de aplicaciones determinadas. En realidad, una elección errónea podría acarrear resultados desastrosos, no sólo con respecto a la eficiencia o rendimiento del sistema, sino también con respecto a la claridad de código de la solución, sobre todo con vistas a futuras modificaciones. A pesar de ello, no existen, hoy día, unas directrices imparciales que ayuden a la elección de un sistema con el objetivo de resolver un problema de satisfacción de restricciones (CSP).

De los dominios para CLP, el dominio finito (FD) (Van Hentenryck, 1989) es uno de los más estudiados puesto que es adecuado para resolver CSPs discretos. Particularmente, FD es muy útil para los problemas de modelado tales como los problemas de

planificación, ordenación, empaquetamiento, programación cronológica (o sea, ordenación temporal) y, por ello, la mayoría de los sistemas de CLP suministran librerías de restricciones sobre el FD. También, como ya se ha apuntado en el Capítulo 1, nosotros estamos principalmente interesados en el enfoque transparente para la resolución de las restricciones y en la práctica nos encontramos con el hecho de que la mayoría de los actuales sistemas transparentes para CLP están diseñados para el dominio finito. Por esta razón, en este capítulo hemos considerado sólo el dominio finito y el Booleano (el cual en realidad es una instancia - con sólo dos valores- del dominio finito). El principal objetivo de la comparación que hemos realizado es el de escoger el mejor enfoque transparente sobre el cual implementar nuestro resolutor genérico. El enfoque escogido tiene que permitir cierta flexibilidad en la formulación de problemas y, también, tiene que mostrar un rendimiento aceptable de los sistemas sobre los cuales esté implementado (i.e., debe combinar declaratividad con eficiencia).

El estudio realizado en este capítulo proporciona una comparativa imparcial de diverso resolutores en el FD para CLP. Hemos comparado ocho sistemas: ECL$^i$PS$^e$ (Aggoun et al., 1995), Oz (Smolka, 1995), Ilog SOLVER (Ilog, 1995), clp(FD) (Codognet y Diaz, 1996a), CHR (Frühwirth, 1998), SICStus (Sicstus manual, 1994), IF/Prolog (If/Prolog, 1994) y B-Prolog (Zhou, 1997). Estos sistemas han sido escogidos porque cubren las principales clases de resolutores sobre el FD y son todos muy populares dentro de la comunidad de la CLP.

Aunque los sistemas han sido probados sobre un conjunto tradicional de problemas clásicos, la mayoría del trabajo comparativo existente en la literatura ha sido realizado por los mismos implementadores de los sistemas siendo comparados (Carlsson et al., 1997; Codognet y Diaz, 1994; Codognet y Diaz, 1996a; Cras, 1993; Müller y Würtz, 1996; Puget y Leconte, 1995; Sidebottom, 1993), lo que supone que los problemas escogidos y la valoración final del sistema podría no ser totalmente imparcial. Nosotros no somos los implementadores, ni pertenecemos al grupo investigador de ninguno de estos sistemas por lo que la comparación que hemos realizado es más imparcial que otras precedentes.

Hemos escogido un tipo particular de puzzle lógico llamado *puzzle auto-referencial* (Self-Referential Quiz, SRQ), como el problema principal sobre el cual hemos realizado la comparativa. Este tipo de puzzles fueron implementados, por primera vez, en (Henz, 1996), con el objetivo de demostrar las capacidades para el meta-razonamiento del sistema Oz. A causa de la naturaleza auto-referencial del problema, este tipo de puzzles es particularmente adecuado para examinar la facilidad con la que los diferentes lenguajes pueden ser usados sobre problemas que requieren meta-razonamiento. Por lo tanto, hemos usado el puzzle SRQ para mostrar cómo los lenguajes soportan la 'reificación de restricciones' y las 'meta-restricciones' (véase la Sección 2.2). También hemos usado este puzzle para comparar la eficiencia de los sistemas.

Sin embargo, se sabe que un único problema podría desviar la evaluación de un sistema de programación y los resultados obtenidos no serían extrapolables globalmente para valorar el sistema. Por ello, para hacer la comparativa más objetiva, hemos extendido nuestro estudio con otros problemas (dos de los cuales son escalables) y hemos comparado, principalmente, la eficiencia de las soluciones. Escogimos problemas

bien conocidos y usamos, siempre que nos fue posible, las soluciones proporcionadas en la distribución del sistema, o en su defecto, formuladas directamente por los mismos implementadores (a través de comunicaciones personales).

En realidad, todos los sistemas evaluados son transparentes. Sin embargo, atendiendo al nivel de transparencia, los hemos catalogado en sistemas transparentes y en sistemas opacos. Así, aquellos sistemas que requieren, por parte del usuario, de un conocimiento detallado de la implementación para crear nuevas restricciones, han sido catalogados como sistemas opacos. En esta categoría tenemos los siguientes sistemas:

- Oz 2.0.
- ECL$^i$PS$^e$ 3.5.2.
- Ilog SOLVER 3.1
- B-Prolog 2.1

Por el contrario, el resto de los sistemas permiten crear nuevas restricciones sin necesidad de tener un conocimiento profundo de la implementación. Estos sistemas son: clp(FD) 2.21, SICStus 3#5, IF/Prolog 5.0 and CHR. Los tres primeros están basados en el enfoque transparente de indexicals sobre el dominio finito (Van Hentenryck et al., 1991; Diaz, 1994; Diaz, 1995; Codognet y Diaz, 1996a). El último está basado en el modelo de reglas de manejo de restricción (Frühwirth, 1994; Frühwirth, 1998; Frühwirth, 1999; Abdennadher et al., 1999; Abdennadher, 1997).

## 2.2. Evaluando la Expresividad

Nosotros sólo evaluamos un aspecto de le expresividad de los lenguajes: la capacidad para expresar restricciones reificadas y meta-restricciones.

### 2.2.1. Restricciones Reificadas

*Una restricción reificada* (RCs) es aquella que refleja la validez de una restricción en una variable Booleana. Este tipo de restricciones permite además que el valor de verdad de una variable Booleana imponga una restricción determinada. En general, las RCs tienen la siguiente forma:

$$b \equiv c$$

donde $b$ toma el valor *true* tan pronto como se conoce que la restricción $c$ es verdadera, y $b$ toma el valor *false* tan pronto como se conoce que la restricción $c$ es falsa. Por otra parte, si $b$ es *true* entonces la restricción $c$ es impuesta, y si $b$ es *false* entonces se impone la negación de $c$.

**EJEMPLO 2.1** *The constraint*

$$x \equiv (y + z > v)$$

*restringe $x$ a true cuando se conoce que inecuación se cumple y $x$ toma el valor false tan pronto como se sabe que la inecuación no se cumplirá jamás. Por otro lado, si restringimos $x$ al valor true entonces la inecuación tiene que cumplirse (es decir, es impuesta) y si $x$ se restringe al valor false entonces la negación de la inecuación tiene que satisfacerse.*

### 2.2.2. Meta-restricciones

Una meta-restricción es una forma de expresar restricciones a partir de otras restricciones. Una forma de expresar una meta-restricción consiste en usar las conectivas lógicas y aplicarlas directamente sobre otra restricción o sobre una expresión que usa estas conectivas lógicas.

**EJEMPLO 2.2** *La siguiente fórmula lógica*

$$(x < 4) \equiv (y < 3) \wedge (z \geq 8) \vee (w \leq 3) \wedge (w = y)$$

*restringe $x$ a ser menor que 4 siempre que $y < 3$ y $z \geq 8$ o siempre que $w \leq 3$ y $w = y$. También, tan pronto como $x$ es menor que 4, la disyunción de restricciones a la derecha de la formula es impuesta (o sea, tiene que satsfacerse para que la fórmula global se satisfaga).*

### 2.2.3. Valoración de los Resultados

A continuación resumimos los resultados de esta comparativa en expresividad.

**Oz, SICStus y IF/Prolog**.

• Estos lenguajes permiten una forma de restricción reificada y admiten la concatenación de propagadores en la forma

$$arg_0\ R_1\ arg_1\ R_2 \ldots R_{n-1}\ arg_{n-1}\ R_n\ arg_n,$$

donde cada $R_i$ es un propagador y $arg_{i-1}$ y $arg_i$ son argumentos ($i \in \{1 \ldots n\}$).

• Los propagadores pueden operar directamente sobre variables Booleanas (i.e. $B_1 \vee B_2$ o $\sim B_1$, donde $B_1$ y $B_2$ son variables Booleanas y $\vee$ y $\sim$ denotan los propagadores para la disyunción y la negación respectivamente[1]) o sobre expresiones (i.e. $(Q_1 = n_1) \vee (Q_2 = n_2)$ donde $Q_1$ y $Q_2$ son variables de dominio finito y los valores $n_1$ y $n_2$ pertenecen al dominio finito).

• Las meta-restricciones pueden ser combinadas con restricciones reificadas como, por ejemplo, en $B \equiv (Q_1 = n_1) \wedge (Q_2 = n_2)$ donde $B$ es una variable Booleana.

**ECL$^i$PS$^e$ y Ilog SOLVER**.

[1] La sintaxis de los propagadores varía de un lenguaje a otro.

• Los propagadores pueden ser concatenados como en Oz, SICStus o IF/Prolog. La principal diferencia con estos lenguajes es que algunos propagadores están limitados a operar sobre expresiones y no directamente sobre las variables (tal como era permitido en Oz, SICStus o IF/Prolog).

• ECL$^i$PS$^e$ no permite el uso de restricciones reificadas.

• Ilog SOLVER posibilita la reificación para las variables Booleanas.

**clp(FD) and B-Prolog**.

• La mayoría de los propagadores tienen una forma relacional (es decir, son de la forma *R(X,Y,Result)*, donde $R$ es un propagador y $X, Y$ y *Result* son variables Booleanas) que está cercana al estilo tradicional de programación de Prolog. Como consecuencia, la concatenación directa de operadores no es posible, y en todo caso, hay que añadir variables adicionales.

• Las meta-restricciones no pueden ser formuladas directamente en clp(FD) y hay que utilizar un interface al lenguaje C para definirlas.

• En B-Prolog, como en clp(FD), las meta-restricciones no pueden ser codificadas directamente y hay que aprender el mecanismo de cláusulas de retraso, que proporciona el sistema, para implementar las meta-restricciones.

### 2.2.4. CHR: una Mención Especial

Expresivamente hablando, el lenguaje CHR merece una consideración especial debido a su flexibilidad para generar los resolutores. Su particular enfoque transparente posibilita que un mismo problema pueda ser formulado de formas muy diferentes, incluso permite que pueda ser codificado usando el estilo de todos y cada uno de los lenguajes considerados en nuestra comparativa.

## 2.3. Evaluando el Rendimiento

En esta sección resumimos los principales resultados obtenidos.

**Ilog SOLVER**. En general, Ilog fue el sistema más rápido con diferencia, Además se mostró extremadamente robusto, pudiendo resolver problemas que involucraban cerca de 1600 variables de dominio finito.

**clp(FD)**. Este sistema mostró buenos resultados, aunque no fue tan rápido como Ilog. Desafortunadamente, devolvía mensajes de error conforme el tamaño del problema aumentaba, indicando que no escala demasiado bien con respecto al número de variables de FD.

**Oz**. Es más rápido que ECL$^i$PS$^e$ (el cual se tomó como referencia) cuando se intenta buscar la primera solución y también cuando se intentan encontrar todas las soluciones. Incluso en los problemas escalables se mostró casi tan bueno, en encontrar todas las

soluciones, como el sistema clp(FD) (con un número bajo de variables). Sin embargo se mostró más robusto que clp(FD).

**SICStus** and **IF/Prolog** tuvieron comportamientos similares y fueron, de dos a tres veces, más rápidos que ECL$^i$PS$^e$ (aunque particularmente IF/Prolog se comportó realmente mal en algún que otro problema). Por otra parte, IF/Prolog mostró un más que aceptable comportamiento en la búsqueda de la primera solución (incluso mejor, a veces, que clp(FD) y Ilog). IF/Prolog y SICStus fueron también más robustos que clp(FD), Oz, y ECL$^i$PS$^e$. Comparando ambos en este aspecto, SICStus fue mejor que IF/Prolog puesto que pudo resolver problemas con cerca de 1000 variables de FD mientras que IF/Prolog sólo pudo resolver problemas con unas 600 variables.

**B-Prolog**. Como clp(FD), este sistema funcionó aceptablemente bien sobre problemas involucrando un número pequeño de variables. Sin embargo, conforme aumentaba el número de variables restringidas sobre el FD, en los problemas escalables, el rendimiento fue deteriorándose y, en la mayoría de los casos, el programa finalizaba devolviendo mensajes de error. Esto es consecuencia directa del hecho de que la versión de este sistema aquí estudiada carece de un recolector de basura.

**ECL$^i$PS$^e$**. Mostró los peores resultados (exceptuando el lenguaje CHR, cuya versión usada está implementada sobre el sistema ECL$^i$PS$^e$). Para obtener un mejor rendimiento tuvimos que desactivar el recolector de basura, lo que a su vez, daba problemas para resolver los problemas escalables. En general, este sistema mostró un rendimiento pobre.

**CHR**. Fue el peor. Apuntamos dos razones principales: (1) el sistema que usamos fue la versión que está construida sobre el sistema ECL$^i$PS$^e$ (el cual a su vez mostró un rendimiento pobre) y (2) CHR no fue diseñado con el objetivo de mejorar la eficiencia de los sistemas, sino con el de definir nuevas restricciones para solucionar problemas específicos sobre dominios específicos.

## 2.4. Conclusiones

Para resumir: eficientemente hablando, el resolutor de Ilog es el mejor. El sistema clp(FD) es también un buen candidato cuando el problema no es demasiado grande (medido en el número de variables restringidas); expresivamente hablando, CHR es el mejor; finalmente para un balance entre la expresividad y la eficiencia, IF/Prolog y SICStus se comportaron razonablemente bien, aunque SICStus mostró una mejor robustez.

## 2.5. Contribuciones

Hasta lo que conocemos, ésta es la primera vez que tantos sistemas de restricciones son comparados sobre el dominio finito desde dos puntos de vista muy diferentes: comparación de la expresividad y de la eficiencia. El capítulo aporta dos contribuciones principales:

- para la comunidad de las restricciones, nuestra experiencia descrita en este capítulo, puede ayudar a otros a escoger el mejor sistema de CLP, sobre el FD, para resolver un problema (discreto) de satisfacción de restricciones y
- para la tesis, los resultados mostrados en este capítulo justifican la elección del modelo de indexicals como el enfoque a partir del cual diseñamos nuestro marco genérico para la resolución de restricciones de intervalo (y que se describe en los capítulos siguientes). Observa que, en general, los sistemas sobre el FD que adoptan este modelo combinan una buena expresividad con un, más que aceptable, rendimiento.

# Parte III

# Marco Teórico

# Capítulo 3

# Propagación de Restricciones de Intervalo sobre Retículos de Intervalo

*[Boswell:] Sir, what is poetry?*
*[Johnson:] Why Sir, it is much easier to say what it is not.*
*We all know what light is*
*but it is not easy to tell what it is.*

Boswell Life, vol.3, pag: 38 (1776)
Samuel Johnson, 1709-84

## 3.1. Motivaciones

Los sistemas de CLP soportan diferentes dominios de computación tales como los intervalos finitos de enteros (Carlson et al., 1994; Carlsson et al., 1997; Codognet y Diaz, 1996a), los reales (Jaffar et al., 1992; Refalo y Van Hentenryck, 1996; Sidebottom y Havens, 1992; Benhamou, 1995), los conjuntos finitos (Walinsky, 1989; Gervet, 1997; Müller y Müller, 1997) o los Booleanos (Codognet y Diaz, 1996b; Codognet y Diaz, 1994; Barth y Bockmayr, 1996). El tipo del dominio determina la naturaleza de las restricciones y el resolutor usado para resolverlas. En particular, la cardinalidad del dominio determina el procedimiento de resolución de tal forma que los actuales sistemas de CLP tienen métodos diferentes para el dominio finito y para los dominios infinitos.

En general, en la práctica los problemas no son específicos a un dominio y su formulación tiene que ser artificialmente adaptada a un resolutor específico (i.e., a un dominio específico). Además, la mayoría de los resolutores de restricción son *opacos* y

su control depende exclusivamente del sistema. Estas restricciones proporcionan implementaciones muy eficientes para aplicaciones prácticas pero carecen de la flexibilidad para ser usados en aplicaciones para las cuales no fueron creados. Este problema es solucionado por las llamadas *restricciones transparentes* (Frühwirth, 1998; Codognet y Diaz, 1996a) las cuales pueden ser definidas por el usuario. Nosotros estamos principalmente interesados en estas últimas, y en particular, en aquellas que no necesitan un conocimiento detallado de la implementación del sistema para ser construidas en el nivel de usuario.

Desde este punto de vista, existen dos sistemas principales que aportan el enfoque transparente sobre las restricciones: el sistema clp(FD) (Codognet y Diaz, 1996a) y las *CHR* (Constraint Handling Rules, i.e., reglas de manejo de las restricciones) (Frühwirth, 1998). El primero de éstos fue diseñado para el dominio finito (FD), y se basa en una restricción simple que se usa para construir restricciones mas complejas y para controlar el mecanismo de propagación. Estas restricciones, a menudo denominadas como *indexicals*, son muy eficientes (tal como se demostró en el Capítulo 2) pues la implementación usa una técnica muy simple de reducción de intervalos que puede ser fácilmente integrada en la máquina abstracta de Warren (WAM) (Aït-kaci, 1999; Diaz y Codognet, 1993). Por ello, clp(FD) forma parte, hoy día, de los principales sistemas de CLP tales como SICStus Prolog (Carlsson et al., 1997) o IF/Prolog (If/Prolog, 1994). Por otro lado, CHR (actualmente incluida como una librería en diversos sistemas (Carlsson et al., 1997; Aggoun et al., 1995; G. y Treinen R., 1995)) facilita la creación de restricciones y resolutores definidos por el usuario y permite además interacción entre ellos. Desafortunadamente, la flexibilidad de estas reglas tiene un coste y, como se mostró en el Capítulo 2, el sistema CHR no puede competir, eficientemente hablando, con otros sistemas que emplean enfoques más tradicionales.

De esta discusión se deduce que lo que se necesita es un sistema transparente que combine la flexibilidad del sistema CHR con la eficiencia del sistema clp(FD). En el capítulo previo se demostró que los sistemas sujetos al enfoque de los indexicals suministraban un buen balance entre expresividad y eficiencia. A partir de este resultado, en este capítulo adoptamos el modelo de indexicals de clp(FD) para la propagación de restricciones y lo generalizamos a cualquier conjunto de retículos, proporcionando por tanto, una flexibilidad cercana a ésa de CHR.

Observa además que nuestra generalización, al ser aplicable a cualquier retículo, es válida para todos los dominios prácticos existentes en CLP (e.g. reales, enteros, conjuntos y Booleanos). Más aún, mediante el uso de operadores de combinación de retículos, pueden obtenerse nuevos dominios de computación y los resolutores pueden generarse a partir de los resolutores ya existentes. Nuestro marco teórico también permite una forma de cooperación en la cual la información fluye entre dominios diferentes a través de canales de sentido único.

Nuestra generalización es nueva incluso en el caso de los dominios finitos de enteros en el cual el modelo de indexicals es ampliamente conocido y estudiado. Nuestra propuesta es también una alternativa al sistema CHR y tiene muchas ventajas potenciales como indicamos a continuación:

- La única condición que hemos impuesto sobre el dominio de computación es que sea un retículo. Puesto que, por lo que sabemos, todos los dominios existentes soportados para CLP, son retículos o podrían ser fácilmente extendidos como retículos, esto quiere decir que nuestro enfoque permite una gran variedad de aplicaciones.

- Existen operadores bien conocidos en la teoría de retículos, tales como el producto directo y el lexicográfico, que permiten combinar retículos. Por lo tanto, es bastante sencillo combinar dominios existentes y los resolutores de éstos para formar nuevos dominios (compuestos).

- El esquema se define para un conjunto de dominios donde se permite la propagación de la información entre los dominios de manera que resolutores definidos sobre dominios distintos pueden comunicarse y, por lo tanto, cooperar.

- El esquema básico para la propagación de restricciones es uniforme para todos los dominios independientemente de si éstos son predefinidos o han sido definidos por el usuario y de la cardinalidad de los mismos.

- Nuestro esquema es un entorno totalmente transparente sobre el cual se pueden definir, a nivel de usuario, nuevos dominios, nuevas restricciones sobre estos dominios y el proceso de propagación entre ellos.

## 3.2. Conceptos Preliminares y Notación

Si $C$ es un conjunto, entonces $\#C$ denota su cardinalidad, $\wp(C)$ su conjunto potencia y $\wp_f(C)$ el conjunto de todos los subconjuntos finitos de $C$, es decir, $\wp_f(C) = \{c \in \wp(C) \mid c \text{ is finite}\}$.

**Orden.** Sea $C$ un conjunto con la igualdad definida. Una relación binaria $\preceq$ sobre $C$ es una *relación de orden* si es reflexiva, antisimétrica y transitiva. Sea $C$ un conjunto con una relación de orden $\preceq$ y donde $c, c' \in C$. Entonces, escribimos $c \preceq_C c'$ (cuando sea necesario) para expresar que $c \preceq c'$ y escribimos $c \sim c'$ si $c \preceq c'$ o $c' \preceq c$ y, en otro caso escribimos $c \not\sim c'$. También $c \prec_C c'$ si $c \preceq_C c'$ y $c' \not\preceq_C c$. Cualquier conjunto $C$ sobre el cual se define una relación de orden se llama *ordenado*. Decimos que $C$ está *totalmente ordenado* si para cualquier $a, b \in C$, $a \sim b$.

**Cotas.** Sea $C$ un conjunto ordenado. Un elemento $c \in C$ es una *cota inferior (superior)* de un subconjunto $E \subseteq C$ si y sólo si $\forall x \in E$: $c \preceq x$ ($x \preceq c$). Si el conjunto de cotas inferiores (superiores) de $E$ tiene un elemento superior (inferior), entonces denominaremos a ese elemento la *c*ota inferior más alta (la *c*ota superior más baja) de $E$ y se denota por $glb_C(E)$ ($lub_C(E)$). Si $E = \{x, y\}$, nosotros escribimos $glb_C(x, y)$ para denotar $glb_C(\{x, y\})$ y $lub_C(x, y)$ para denotar $lub_C(\{x, y\})$.

**Predecesor y sucesor.** Sea $C$ un conjunto totalmente ordenado y sea $c, c' \in C$. Entonces $c$ es el *predecesor inmediato* de $c'$ y $c'$ el *sucesor inmediato* de $c$ si $c \prec c'$ y para cualquier $c'' \in C$ con $c \preceq c'' \prec c'$ implica $c = c''$.

**Monotonía.** Sea $f$ una función n-aria $f :: C_1 \times \ldots \times C_n \to C$, donde $C$ y cada $C_i$, para $i \in \{1, \ldots, n\}$, son conjuntos ordenados. Entonces $f$ es *monótona* si, para todo $i \in \{1, \ldots, n\}$ tal que $t_i, t_i' \in C_i$ y $t_i \preceq_{C_i} t_i'$, entonces

$$f(t_1, \ldots, t_i, \ldots, t_n) \preceq_C f(t_1', \ldots, t_i', \ldots, t_n').$$

y *anti-monótona* si

$$f(t_1, \ldots, t_i, \ldots, t_n) \succeq_C f(t_1', \ldots, t_i', \ldots, t_n').$$

Una función monótona $f$ es *estrictamente monótona* cuando para todo $i \in \{1, \ldots, n\}$, tal que $t_i, t_i' \in C_i$, $t_i \preceq_{C_i} t_i'$ y existe algún $j \in \{1, \ldots, n\}$ tal que $t_j \prec_{C_j} t_j'$, entonces

$$f(t_1, \ldots, t_i, \ldots, t_n) \prec_C f(t_1', \ldots, t_i', \ldots, t_n').$$

y es *estrictamente monótona* si

$$f(t_1, \ldots, t_i, \ldots, t_n) \succ_C f(t_1', \ldots, t_i', \ldots, t_n').$$

**Retículo.** Sea $L$ un conjunto ordenado. $L$ es un *retículo* si $lub_L(x, y)$ y $glb_L(x, y)$ existen para cada par de elementos $x, y \in L$.

**Elementos top y bottom.** Sea $L$ un retículo. Si existe, $glb_L(L) = \bot_L$ es *el elemento mínimo (bottom)* de $L$. De forma similar, si existe, $lub_L(L) = \top_L$ es *el elemento máximo (top)* de $L$. La carencia de un elemento top o bottom puede ser remediado añadiendo una cota ficticia. Por lo tanto, el *retículo liftado* de $L$ es $L \cup \{\bot_L, \top_L\}$ donde, si $glb_L(L)$ no existe, $\bot_L$ es un nuevo elemento no perteneciente a $L$ tal que $\forall a \in L, \bot_L \prec a$ y similarmente, si $lub_L(L)$ no existe, $\top_L$ es un nuevo elemento no perteneciente a $L$ tal que $\forall a \in L, a \prec \top_L$.

**Dual.** Sea $L$ un retículo. El *dual de* $L$, denotado por $\hat{L}$, es el retículo que contiene exactamente los mismo elementos que $L$ y que se obtiene intercambiando $glb_L(a, b)$ y $lub_L(a, b)$ para cualquier $a, b \in L$. Si $a \in L$, entonces denotamos su dual como $\hat{a} \in \hat{L}$. El *principio de dualidad de los retículos* es "el dual de una sentencia sobre retículos expresada en términos de *glb* y *lub* puede ser obtenido simplemente intercambiando *glb* y *lub*". Observa que el orden se invierte en el retículo dual. Es decir, si $a, b \in L$ y $a \preceq_L b$, entonces $\hat{b} \preceq_{\hat{L}} \hat{a}$.

**Productos.** Sean $L_1$ y $L_2$ dos retículos (liftados). Entonces el producto directo $\langle L_1, L_2\rangle$ y el producto lexicográfico $(L_1, L_2)$ son retículos tales que:

$$\langle x_1, x_2\rangle \preceq_{\langle L_1,L_2\rangle} \langle y_1, y_2\rangle \text{ iff } x_1 \preceq_{L_1} y_1 \text{ y } x_2 \preceq_{L_2} y_2;$$
$$(x_1, x_2) \preceq_{(L_1,L_2)} (y_1, y_2) \text{ iff } x_1 \prec_{L_1} y_1 \text{ o } (x_1 = y_1 \text{ y } x_2 \preceq_{L_2} y_2).$$

Más aún

$$\begin{aligned} glb(\langle x_1, x_2\rangle, \langle y_1, y_2\rangle) = &\quad \langle glb_{L_1}(x_1, y_1), glb_{L_2}(x_2, y_2)\rangle; \\ glb((x_1, x_2), (y_1, y_2)) = &\quad \text{si } x_1 = y_1 \text{ entonces } (x_1, glb_{L_2}(x_2, y_2)) \\ &\quad \text{sino si } x_1 \prec y_1 \text{ entonces } (x_1, x_2) \\ &\quad \text{sino si } x_1 \succ y_1 \text{ entonces } (y_1, y_2) \\ &\quad \text{sino } (glb_{L_1}(x_1, y_1), \top_{L_2}); \end{aligned}$$

*lub* es el dual de *glb* and

$$\begin{aligned} \top_{\langle L_1,L_2\rangle} &= \langle \top_{L_1}, \top_{L_2}\rangle \text{ y } \bot_{\langle L_1,L_2\rangle} = \langle \bot_{L_1}, \bot_{L_2}\rangle; \\ \top_{(L_1,L_2)} &= (\top_{L_1}, \top_{L_2}) \text{ Y } \bot_{(L_1,L_2)} = (\bot_{L_1}, \bot_{L_2}). \end{aligned}$$

**Sumas Lineales.** Supón que $L_1, \ldots, L_n$ son retículos. Entonces su *suma lineal*, denotada como $L_1 \uplus \ldots \uplus L_n$, es el retículo $L_S$ donde:
(1) $L_S = L_1 \cup \ldots \cup L_n$;
(2) la relación de orden $\preceq_{L_S}$ se define:

$$x \preceq_{L_S} y \iff \begin{cases} x, y \in L_i \text{ y } x \preceq_{L_i} y; \text{ o} \\ x \in L_i, y \in L_j \text{ y } i \prec j; \end{cases}$$

(3) $glb_{L_S}$ y $lub_{L_S}$ se definen:

$$\begin{aligned} glb_{L_S}(x, y) = &\quad glb_{L_i}(x, y) \text{ y } lub_{L_S}(x, y) = lub_{L_i}(x, y) \text{ si } x, y \in L_i \\ glb_{L_S}(x, y) = &\quad x \text{ y } lub_{L_S}(x, y) = y \text{ si } x \in L_i,\ y \in L_j \text{ y } i \prec j \\ glb_{L_S}(x, y) = &\quad y \text{ y } lub_{L_S}(x, y) = x \text{ si } x \in L_i,\ y \in L_j \text{ y } j \prec i; \end{aligned}$$

(4) $\bot_{L_S} = \bot_{L_1}$ y $\top_{L_S} = \top_{L_n}$.

Para más información sobre retículos, véase (Davey y Priestley, 1990). En el resto del documento, $(L, \preceq_L, glb_L, lub_L, \bot_L, \top_L)$ denota un retículo (posiblemente liftado) sobre $L$ con cotas $\bot_L$ y $\top_L$ (posiblemente ficticias).

## 3.3. Los Dominios de Computación y el de Intervalo

El dominio sobre el cual los valores se calculan se llama *dominio de computación*. El aspecto clave del sistema de restricciones descrito en este documento es que puede

ser construido sobre cualquier dominio con estructura de retículo. A través de este documento, nosotros denotamos por $\mathcal{L}$ un conjunto (posiblemente infinito) de dominios de computación conteniendo al menos un elemento $L$. También denotamos $\hat{\mathcal{L}} = \{\hat{L} \mid L \in \mathcal{L}\}$. Para cada dominio de computación $L \in \mathcal{L}$, nosotros asociamos un conjunto de símbolos de variables $V_L$ que es disjunto en $\mathcal{L}$. Definimos $\mathcal{V}_{\mathcal{L}} = \cup\{V_L | L \in \mathcal{L}\}$ y asumimos (sin pérdida de generalidad) que cada $L \in \mathcal{L}$ es un retículo liftado.

**EJEMPLO 3.1** *La mayoría de los dominios de computación clásicos son retículos. Por ejemplo,*

$$(Integer, \leq, mini, maxi, \bot_{Integer}, \top_{Integer}),$$
$$(\Re, \leq, mini, maxi, \bot_{\Re}, \top_{\Re}),$$
$$(Bool, \leq, \wedge, \vee, false, true),$$
$$(Set\ L, \subseteq, \cap, \cup, \emptyset, L)$$

*son retículos para los enteros, los reales, los Booleanos y los conjuntos, respectivamente, bajo las relaciones de orden convencionales, donde las funciones 'mini' y 'maxi' devuelven, respectivamente, el mínimo y el máximo de dos elementos enteros o reales. Observa que Integer y $\Re$ son retículos liftados que incluyen los elementos ficticios $\top_{Integer}$, $\bot_{Integer}$, $\top_{\Re}$ y $\bot_{\Re}$. Para los Booleanos, se supone que $Bool = \{false, true\}$ y $false <_{Bool} true$. Para el retículo conjunto (i.e., Set ) suponemos que*

$$Set\ \ L = \wp(L),$$

*para cada $L \in \mathcal{L}$, donde $\mathcal{L} = \{Integer, \Re, Bool\} \cup \{Set\ \ L \mid L \in \mathcal{L}\}$. Observe que $\mathcal{L}$ es un conjunto infinito de dominios de computación.*

En el resto de este documento haremos referencia a estos dominios de computación *Integer*, $\Re$, *Bool* y *Set* $L$, para cualquier dominio $L \in \mathcal{L}$, sin hacer ninguna referencia a este ejemplo.

### 3.3.1. Dominios de Computación Acotados

Primero definimos el dominio de los paréntesis.

**DEFINICIÓN 3.2** *(Dominio paréntesis) El* dominio de los paréntesis $B$ *es un retículo que contiene sólo dos elementos* ')' *y* ']' *donde* ')' $\prec_B$ ']'.

En el resto del documento, '}' denota cualquier elemento de $B$. También definimos una función $min_B(\}_1, \}_2)$ que devuelve el menor elemento entre $\}_1$ y $\}_2$ en $B$.

Este dominio se combina con cualquier dominio de computación para formar la cota derecha de un intervalo.

**DEFINICIÓN 3.3** *(El dominio acotado y simple de computación) El* dominio acotado y simple de computación *(para $L$), denotado por $L^s$, es el producto lexicográfico* $(L, B)$.

Ahora definimos el *reflejo* de este dominio para formar la cota izquierda de un intervalo.

**DEFINICIÓN 3.4** *(El dominio reflejo) El* espejo (de $L^s$) *es el producto lexicográfico* $(\hat{L}, B)$*, denotado por* $\overline{L^s}$*. EL* reflejo *de un elemento t=(a,'}') ∈* $L^s$ *es el elemento* $(\hat{a}, \text{'}\}\text{'}) \in \overline{L^s}$ *y lo denotamos como* $\bar{t}$*.*

**PROPOSICIÓN 3.5**

1. $\overline{L^s}$ *es el dominio acotado y simple de computación para* $\hat{L}$*. i.e.,* $\overline{L^s} = \hat{L}^s$*;*
2. $\bot_{\overline{L^s}} = \overline{\top_L)} = (\top_L$ *y* $\top_{\overline{L^s}} = \overline{\bot_L]} = [\bot_L$*.*
3. *Si* $t_1 = (a_1, \text{'}\}_1\text{'}), t_2 = (a_2, \text{'}\}_2\text{'}) \in L^s$ *entonces,*

$$\textit{si } a_1 \neq a_2, \left\{ \begin{array}{c} \overline{t_2} \prec_{\overline{L^s}} \overline{t_1} \iff t_1 \prec_{L^s} t_2 \\ glb_{\overline{L^s}}(\overline{t_1}, \overline{t_2}) = \overline{lub_{L^s}(t_1, t_2)} \\ lub_{\overline{L^s}}(\overline{t_1}, \overline{t_2}) = \overline{glb_{L^s}(t_1, t_2)} \end{array} \right\} \textit{ sino } \left\{ \begin{array}{c} \overline{t_1} \preceq_{\overline{L^s}} \overline{t_2} \iff t_1 \preceq_{L^s} t_2 \\ glb_{\overline{L^s}}(\overline{t_1}, \overline{t_2}) = \overline{glb_{L^s}(t_1, t_2)} \\ lub_{\overline{L^s}}(\overline{t_1}, \overline{t_2}) = \overline{lub_{L^s}(t_1, t_2)} \end{array} \right\}.$$

En el resto de este documento, para simplificar la notación, a menudo denotamos un elemento (a,'}') de $L^s$ como **a}** y un elemento (â,'}') de $\overline{L^s}$ como **{a**. En particular, con esta notación tenemos $\bot_{L^s} = \bot_L$**)**, y $\top_{L^s} = \top_L$**]**.

**EJEMPLO 3.6** *Considera el dominio de computación L = Integer. Entonces* **6]** *denota* $(6, \text{'}]\text{'}) \in Integer^s$*,* [**6** *denota* $(\hat{6}, \text{'}]\text{'}) \in \overline{Integer^s}$ *y* $\overline{\mathbf{6}]} = [\mathbf{6}$*. También*

$$\begin{array}{l} glb_{L^s}(\mathbf{3}], \mathbf{5}]) = lub_{L^s}(\mathbf{3}], \mathbf{3})) = \mathbf{3}], \\ glb_{\overline{L^s}}([\mathbf{3}, [\mathbf{5}) = \overline{lub_{L^s}(\mathbf{3}], \mathbf{5}])} = \overline{\mathbf{5}]} = [\mathbf{5}, \\ lub_{\overline{L^s}}([\mathbf{3}, [\mathbf{5}) = \overline{glb_{L^s}(\mathbf{3}], \mathbf{5}])} = \overline{\mathbf{3}]} = [\mathbf{3}, \\ \mathbf{0}) \prec_{L^s} \mathbf{0}] \prec_{L^s} \mathbf{1}) \prec_{L^s} \mathbf{1}] \prec_{L^s} \ldots \prec_{L^s} \top_L) \ \prec_{L^s} \top_L] \textit{ en } L^s, \\ (\top_L \prec_{\overline{L^s}} [\top_L \prec_{\overline{L^s}} \ldots \prec_{\overline{L^s}} (\mathbf{1} \prec_{\overline{L^s}} [\mathbf{1} \prec_{\overline{L^s}} (\mathbf{0} \prec_{\overline{L^s}} [\mathbf{0}. \end{array}$$

*Más aún, si L = Set Integer, entonces* $\{\mathbf{1}, \mathbf{3}\})$ *denota* $(\{1,3\}, \text{'}) \text{'})$*,* $(\{\mathbf{1}, \mathbf{3}\}$ *denota* $(\widehat{\{1,3\}}, \text{'}) \text{'})$ *y* $\overline{\{\mathbf{1}, \mathbf{3}\}]} = [\{\mathbf{1}, \mathbf{3}\}$*. También*

$$\begin{array}{l} glb_{L^s}(\{\mathbf{4}\}], \{\mathbf{4}, \mathbf{6}\}]) = \{\mathbf{4}\}] \textit{ y } lub_{L^s}(\{\mathbf{3}\}], \{\mathbf{4}, \mathbf{6}\}]) = \{\mathbf{3}, \mathbf{4}, \mathbf{6}\}], \\ glb_{\overline{L^s}}([\{\mathbf{4}\}, [\{\mathbf{4}, \mathbf{6}\}) = \overline{lub_{L^s}(\{\mathbf{4}\}], \{\mathbf{4}, \mathbf{6}\}])} = \overline{\{\mathbf{4}, \mathbf{6}\}]} = [\{\mathbf{4}, \mathbf{6}\}, \\ lub_{\overline{L^s}}([\{\mathbf{4}\}, [\{\mathbf{4}, \mathbf{6}\}) = \overline{glb_{L^s}(\{\mathbf{4}\}], \{\mathbf{4}, \mathbf{6}\}])} = \overline{\{\mathbf{4}\}]} = [\{\mathbf{4}\}, \\ \{\mathbf{1}\}] \prec_{L^s} \{\mathbf{1}, \mathbf{3}\}) \prec_{L^s} \{\mathbf{1}, \mathbf{3}\}] \prec_{L^s} \{\mathbf{1}, \mathbf{3}, \mathbf{5}\}], \\ [\{\mathbf{1}, \mathbf{3}, \mathbf{5}\} \prec_{\overline{L^s}} (\{\mathbf{1}, \mathbf{3}\} \prec_{\overline{L^s}} [\{\mathbf{1}, \mathbf{3}\} \prec_{\overline{L^s}} [\{\mathbf{1}\}. \end{array}$$

**PROPOSICIÓN 3.7** *Sean* $L' \in \{L^s, \overline{L^s}\}$ *y* $t \in L'$*. Entonces,*

$$(1)\ \overline{\overline{L'}} = L', \quad (2)\ \overline{L'} \in \{L^s, \overline{L^s}\}, \quad (3)\ \bar{t} \in \overline{L'} \quad y \quad (4)\ \bar{\bar{t}} = t.$$

### 3.3.2. Operadores de Restricción

Extendemos los dominios de computación acotados para permitir operadores sobre los dominios de computación.

**DEFINICIÓN 3.8** *(Operadores de restricción) Supón que* $L, L_1 \ldots, L_n \in \mathcal{L} \cup \hat{\mathcal{L}}$ *y que* $\circ_L$ *y* $\circ_B$ *son funciones n-arias monótonas definidas como*

$$\circ_L :: L_1 \times \ldots \times L_n \to L,$$
$$\circ_B :: \underbrace{B \times \ldots \times B}_{n \text{ times}} \to B.$$

*Supón también que, si* $\circ_L$ *no es monótona estricta, entonces* $\circ_B$ *es una función constante. Sea* $\circ$ *el operador*

$$\circ :: L_1^s \times \ldots \times L_n^s \to L^s$$
$$\circ((a_1, \}_1), \ldots, (a_n, \}_n)) = (\ \circ_L(a_1, \ldots, a_n), \circ_B(\}_1, \ldots, \}_n)\ ). \tag{3.1}$$

*Entonces* $\circ$ *es* un operador de restricción para $L^s$.
*El* espejo $\overline{\circ}$ *de* $\circ$ *se define como*

$$\overline{\circ} :: \overline{L_1^s} \times \ldots \times \overline{L_n^s} \to \overline{L^s}$$
$$\overline{\circ}(\overline{t_1}, \ldots, \overline{t_i}, \ldots, \overline{t_n}) = \overline{\circ(t_1, \ldots, t_i, \ldots, t_n)}. \tag{3.2}$$

Si $\circ$ es un operador binario, permitimos la notación infija de tal forma que la expresión $\circ(t_1, t_2)$ se denota como $t_1 \circ t_2$.

**EJEMPLO 3.9** *Supón que* $+_L$ *y* $-_L$ *se declaran como*

$$+_L :: L \times L \to L \qquad\qquad -_L :: L \times \hat{L} \to L.$$

*Así, si* $L \in \{Integer, \Re\}$ *y los operadores* $+_L/2$ *y* $-_L/2$ *son definidos como es usual y devuelven la suma y la diferencia de dos números en L, estos operadores son monótonos estrictamente. Observa que conforme a y* $\hat{b}$ *aumentan en L y* $\hat{L}$*, respectivamente,* $a -_L \hat{b}$ *también incrementa en L.*

*Supón, más generalmente, que* $+_L/2$ *y* $-_L/2$ *son definidos para un dominio L de tal forma que son monótonos estrictos. Supón también que* $+_B$ *y* $-_B$ *se definen como*

$$\}_1 +_B \}_2 = min_B(\}_1, \}_2) \qquad y \qquad \}_1 -_B \}_2 = ]\ si\ \}_1 =_B \}_2\ y\ )\ en\ otro\ caso.$$

*Entonces los operadores de restricción binarios* $+$ *y* $-$ *para* $L^s$ *y sus operadores reflejos son declarados como*

$$+ :: L^s \times L^s \to L^s \qquad - :: L^s \times \overline{L^s} \to L^s$$
$$\overline{+} :: \overline{L^s} \times \overline{L^s} \to \overline{L^s} \qquad \overline{-} :: \overline{L^s} \times L^s \to \overline{L^s}.$$

*Considera el caso* $L = \Re$*. Entonces tenemos*

$$\mathbf{3},\mathbf{2}) + \mathbf{4},\mathbf{1}] = \mathbf{7},\mathbf{3}), \qquad (\mathbf{3},\mathbf{2}\,\overline{+}\,[\mathbf{4},\mathbf{1} = \overline{\mathbf{3},\mathbf{2}) + \mathbf{4},\mathbf{1}}] = \overline{\mathbf{7},\mathbf{3})} = (\mathbf{7},\mathbf{3},$$
$$\mathbf{7},\mathbf{3}) - [\mathbf{4},\mathbf{1} = \mathbf{3},\mathbf{2}), \qquad (\mathbf{7},\mathbf{3}\,\overline{-}\,\mathbf{4},\mathbf{1}] = \overline{\mathbf{7},\mathbf{3}) - [\mathbf{4},\mathbf{1}} = \overline{\mathbf{3},\mathbf{2})} = (\mathbf{3},\mathbf{2}.$$

**PROPOSICIÓN 3.10** *Supón que* $\circ$ *es un operador de restricción para* $L^s$*. Entonces,*

$\circ$ *es monótono;* (a)

$\overline{\circ}$ *es un operador de restricción (para* $\overline{L^s}$*);* (b)

$\circ$ *es el reflejo de* $\overline{\circ}$ *i.e.,* $\circ \equiv \overline{\overline{\circ}}$*.* (c)

La definición de un operador puede realizarse sobre dominios muy diferentes creando pues un canal de comunicación entre resolutores distintos.

**EJEMPLO 3.11** *Supón que el operador unario trunc se define como*

$$trunc :: \Re^s \to Integer^s$$
$$trunc(\boldsymbol{a}\}) = trunc_{Integer}(\boldsymbol{a})\ trunc_B(])$$

*donde* $trunc_{Integer}(a)$ *devuelve la parte entera de a, para cualquier* $a \in \Re$*, y* $trunc_B(\}) =]$ *para cualquier* $\} \in B$*. Entonces*

$$\overline{trunc} :: \overline{\Re^s} \to \overline{Integer^s}.$$

*Por tanto, por (3.1) y (3.2),*

$$trunc(\mathbf{3},\mathbf{1}]) \ = \ trunc_{Integer}(3,1) trunc_B(]) \ = \ \mathbf{3}],$$
$$\overline{trunc}([\mathbf{3},\mathbf{1}) \ = \ \overline{trunc(\mathbf{3},\mathbf{1}])} \ = \ \overline{\mathbf{3}]} \ = \ [\mathbf{3}.$$

### 3.3.3. Indexicals

Las variables de $V_L$ son introducidas en el dominio $L^s$ y su reflejo mediante *indexicals*. Sea $O_L$ un conjunto de operadores de restricción definido para $L^s$.

**DEFINICIÓN 3.12** *(Dominio de computación acotado) El* dominio de computación acotado $L^b$ *(para L) y su reflejo* $\overline{L^b}$ *se definen de la siguiente manera:*

$$\begin{aligned} L^b = L^s & \\ & \cup \{max(x) \mid x \in V_L\} \\ & \cup \{val(x) \mid x \in V_L\} \\ & \cup \left\{ \circ(t_1, \ldots, t_n) \;\middle|\; \begin{array}{l} \circ :: L'_1 \times \ldots \times L'_n \to L^s \in O_L, \\ t_i \in (L'_i)^b \text{ para } (1 \le i \le n) \end{array} \right\}, \\ \overline{L^b} = \{\overline{t} \mid t \in L^b\} & \end{aligned}$$

*donde para* $i \in \{1, \ldots, n\}$*,* $L_i \in \mathcal{L}$ *y* $L'_i \in \{L^s_i, \overline{L^s_i}\}$*,*

$(L'_i)^b = L^b_i$ *si* $L'_i = L^s_i$ *y* $(L'_i)^b = \overline{L^b_i}$ *si* $L'_i = \overline{L^s_i}$*;*

$\overline{max(x)} = min(x)$*,* $\overline{min(x)} = max(x)$ *y* $\overline{\overline{val(x)}} = val(x)$*;*

*para cada* $\circ :: L'_1 \times \ldots \times L'_n \to L^s \in O_L$,

$$\overline{\circ(t_1, \ldots, t_i, \ldots, t_n)} = \overline{\circ}(\overline{t_1}, \ldots, \overline{t_i}, \ldots, \overline{t_n}).$$

*Las expresiones* $max(x), min(x), val(x)$ *y* $\overline{val(x)}$ *son llamadas* indexicals. *Los elements de* $L^b$ *y* $\overline{L^b}$ *son llamados* términos indexical.

*El dominio* $L^b$ *es también un retículo que hereda el orden de* $L^s$ *y donde* $\top_{L^b} = \top_{L^s}$ *y* $\bot_{L^b} = \bot_{L^s}$. *Por tanto, si* $t_1, t_2 \in L^b$, *entonces* $t_1 \preceq_{L^b} t_2$ *si y sólo si* $t_1 = \bot_{L^b}$, $t_2 = \top_{L^b}$ *o* $t_1, t_2 \in L^s \setminus \{\bot_{L^s}, \top_{L^s}\}$ *y* $t_1 \preceq_{L^s} t_2$.

**PROPOSICIÓN 3.13**

(1) $\overline{\overline{t}} = t$ *para cualquier* $t \in \{L^b, \overline{L^b}\}$.

(2) *Si* $t \in L^b$ *entonces* $\overline{t} \in \overline{L^b}$ *y si* $t \in \overline{L^b}$ *entonces* $\overline{t} \in L^b$.

**EJEMPLO 3.14** *Sean* $+$ *(para* $L = Integer$*),* $-$ *(para* $L = \Re$*) y trunc tal como se definieron en los Ejemplos 3.9 y 3.11. Entonces* $+, trunc \in O_{Integer}$ $y$ $- \in O_{\Re}$. *Sean también* $x \in V_{Integer}$, $y \in V_{\Re}$, *entonces:*

| | | | | |
|---|---|---|---|---|
| **3**], | $max(x)$, | **3**) $+ max(x)$, | $trunc(max(y))$ | *pertenecen a* $Integer^b$*;* |
| **20**,**1**), | $max(y)$, | **20**,**1**) $- min(y)$, | $val(y)$ | *pertenecen a* $\Re^b$*;* |
| [**3**, | $min(x)$, | (**3**$\overline{+}$$min(x)$, | $\overline{trunc}(min(y))$ | *pertenecen a* $\overline{Integer^b}$*;* |
| (**20**,**1**, | $min(y)$, | (**20**,**1**$\overline{-}$$max(y)$, | $\overline{val(y)}$ | *pertenecen a* $\overline{\Re^b}$ |

*donde, por ejemplo,* $\overline{\mathbf{3}) + max(x)} = \overline{\mathbf{3})}\overline{+}\overline{max(x)} = (\mathbf{3}\overline{+}min(x)$.

### 3.3.4. El Dominio de Intervalos

Ahora definimos la estructura sobre la cual las restricciones pueden ser genéricamente resueltas.

**DEFINICIÓN 3.15** *(Dominio de Intervalos) Sea* $L^b$ *el dominio de computación acotado para un dominio* $L \in \mathcal{L}$. *El* dominio de intervalos $R^b_L$ *sobre* $L$ *es el producto directo* $\langle \overline{L^b}, L^b \rangle$. *El* dominio de intervalos simples $R^s_L$ *sobre* $L$ *es el producto directo* $\langle \overline{L^s}, L^s \rangle$. *Un* rango *es un elemento de* $R^b_L$. *Un rango es* simple *si es un elemento de* $R^s_L$.

Por el producto directo de retículos, $R^b_L$ y $R^s_L$ son retículos. Para simplificar la notación, un elemento $\langle \overline{\mathbf{a}\}}, \mathbf{b}\} \rangle$ será escrito simplemente como **{a,b}**.

Por la tanto, para cualquier $r_1 = \langle \overline{s_1}, t_1 \rangle, r_2 = \langle \overline{s_2}, t_2 \rangle \in R^b_L$ donde $s_1, s_2, t_1, t_2 \in L^b$,

$$r_1 \preceq_{R^b_L} r_2 \iff (\overline{s_1} \preceq_{\overline{L^b}} \overline{s_2}) \text{ y } (t_1 \preceq_{L^b} t_2),$$
$$glb_{R^b_L}(r_1, r_2) = \langle glb_{\overline{L^b}}(\overline{s_1}, \overline{s_2}), glb_{L^b}(t_1, t_2) \rangle,$$
$$lub_{R^b_L}(r_1, r_2) = \langle lub_{\overline{L^b}}(\overline{s_1}, \overline{s_2}), lub_{L^b}(t_1, t_2) \rangle,$$
$$\top_{R^b_L} = [\bot_L, \top_L] \text{ y } \bot_{R^b_L} = (\top_L, \bot_L).$$

**EJEMPLO 3.16**

$$\langle \overline{\mathbf{1}}], \mathbf{8}) \rangle \in R^s_{Integer} \textit{ se escribe como } [\mathbf{1}, \mathbf{8}),$$
$$[\mathbf{2},\mathbf{3}, \mathbf{8},\mathbf{9}) \in R^s_{\Re} \quad y \quad [\mathbf{1},\mathbf{4},max(x)+\mathbf{4},\mathbf{9}] \in R^b_{\Re},$$
$$glb_{R^b_{\Re}}([\mathbf{3},\mathbf{2}, \mathbf{6},\mathbf{7}], (\mathbf{1},\mathbf{8}, \mathbf{4},\mathbf{5}]) = [\mathbf{3},\mathbf{2}, \mathbf{4},\mathbf{5}],$$
$$lub_{R^b_{\Re}}([\mathbf{3},\mathbf{2}, \mathbf{6},\mathbf{7}], (\mathbf{1},\mathbf{8}, \mathbf{4},\mathbf{5}]) = (\mathbf{1},\mathbf{8}, \mathbf{6},\mathbf{7}],$$
$$[\mathbf{3},\mathbf{0}, \mathbf{4},\mathbf{0}) \preceq_{R^b_{\Re}} (\mathbf{1},\mathbf{8}, \mathbf{4},\mathbf{5}].$$

*Observe que*

$$glb_{R^b_{\Re}}([\mathbf{3},\mathbf{2}, \mathbf{6},\mathbf{7}], (\mathbf{1},\mathbf{8}, \mathbf{4},\mathbf{5}]) = \langle\ glb_{\overline{\Re^b}}(\ [\mathbf{3},\mathbf{2}, (\mathbf{1},\mathbf{8}\ ), glb_{\Re^b}(\ \mathbf{6},\mathbf{7}], \mathbf{4},\mathbf{5}]\ )\ \rangle = [\mathbf{3},\mathbf{2}, \mathbf{4},\mathbf{5}];$$
$$lub_{R^b_{\Re}}([\mathbf{3},\mathbf{2}, \mathbf{6},\mathbf{7}], (\mathbf{1},\mathbf{8}, \mathbf{4},\mathbf{5}]) = \langle\ lub_{\overline{\Re^b}}(\ [\mathbf{3},\mathbf{2}, (\mathbf{1},\mathbf{8}\ ), lub_{\Re^b}(\ \mathbf{6},\mathbf{7}], \mathbf{4},\mathbf{5}]\ )\ \rangle = (\mathbf{1},\mathbf{8}, \mathbf{6},\mathbf{7}].$$

Es importante resaltar que la relación $\preceq_{R^b_L}$ simula la inclusión de intervalos en la teoría clásica de conjuntos[1]. La Figura 3.1 muestra la relación entre dos elementos cualesquiera de cualquier dominio $R^s_L$ definido sobre algún dominio de computación $L \in \mathcal{L}$.

Figura 3.1: Relación entre dos rangos $\overline{s_1}, t_1$ y $\overline{s_2}, t_2$ de $R^s_L$ para algún $L \in \mathcal{L}$

La consistencia de los rangos simples puede ser genéricamente definida para cualquier dominio de intervalo $R^s_L$ independientemente del retículo $L$.

**DEFINICIÓN 3.17** *(Condiciones de consistencia) Un rango simple* $r = \langle \overline{s}, t \rangle \in R^s_L$ *es* inconsistente *si*

[1]Observe que el concepto de intervalo que usamos en esta tesis difiere del concepto clásico en la teoría de conjuntos.

1. $s \not\preceq_{L^s} t$ *(lo cual quiere decir que* $r$ *es inconsistente si* $s \not\prec_{L^s} t$*) o*
2. $\overline{s} = (\boldsymbol{a}$ *y* $t = \boldsymbol{a}\}$ *para cualquier* $\} \in B$.

*En cualquier otro caso* $r$ *es* consistente. *Observa que* $\bot_{R_L^b} = \bot_{R_L^s} = \langle \bot_{\overline{L^s}}, \bot_{L^s} \rangle = (\top_L, \bot_L)$ *es inconsistente.*

**EJEMPLO 3.18** *En el dominio Integer,* $(\mathbf{1}, \mathbf{1}]$ *y* $[\mathbf{5}, \mathbf{2}]$ *son inconsistentes mientras que* $[\mathbf{1}, \mathbf{10}]$ *es consistente. En el dominio Set Integer,* $[\{\mathbf{1}, \mathbf{3}\}, \{\mathbf{1}\}]$ *y* $[\{\mathbf{1}, \mathbf{3}\}, \{\mathbf{1}, \mathbf{4}\}]$ *son inconsistentes y* $[\{\mathbf{1}\}, \{\mathbf{1}, \mathbf{3}\}]$ *es consistente ya que* $\{\mathbf{1}\}] \prec_{(Set\ Integer)^s} \{\mathbf{1}, \mathbf{3}\}]$.

Cualquier rango (in)consistente en $R_L^s$ identifica un conjunto de rangos (in)consistentes en $R_L^s$.

**PROPOSICIÓN 3.19** *Supón que* $r, r' \in R_L^s$*, para cualquier* $L \in \mathcal{L}$*, donde* $r \preceq_{R_L^s} r'$. *Si* $r'$ *es inconsistente, entonces* $r$ *es también inconsistente.*

**EJEMPLO 3.20** *Supón que* $L \in \mathcal{L}$ *y* $a, b, c \in L$ *donde* $a \prec_L c \prec_L b$. *La Figura 3.2 muestra la zona del retículo* $R_L^s$ *que se construye a partir de los elementos* $a, b$ *y* $c$. *Nota que todos los nodos dentro del cuadrado son inconsistentes mientras que el resto son consistentes. Los nodos marcados con círculos corresponden a casos especiales y son tratados en la Sección 3.4.4.*

## 3.4. Los Dominios de las Restricciones

Las restricciones son expresadas asociando cada variable restringida con uno o más intervalos de valores (definidos usando el dominio de intervalos) sobre el cual (los cuales) la variable puede tomar valores (o sea, está limitada o restringida). Estas *restricciones de intervalo*, las cuales se definen en 3.4.1, son resueltas mediante dos procesos, definidos, respectivamente, en 3.4.2 y 3.4.3: *la estabilización de las restricciones* (i.e., la reducción de las restricciones) y la *propagación de las restricciones* .

Continuamos usando $L$ para denotar cualquier dominio en $\mathcal{L}$, $V_L$ el conjunto de variables asociadas con $L$. También denotamos por $R_L^b$ el dominio de intervalos sobre $L$, $\mathcal{V}_\mathcal{L} = \cup\{V_L | L \in \mathcal{L}\}$ y $X \in \wp_f(\mathcal{V}_\mathcal{L})$.

### 3.4.1. Restricciones de Intervalo

Las *restricciones de intervalo* asocian elementos de $R_L^b$ con variables en $V_L$.

**DEFINICIÓN 3.21** *(El dominio de las restricciones de intervalo) Sean* $x \in V_L$ *y* $r \in R_L^b$. *Entonces*

$$x \sqsubseteq r$$

*es una* restricción de intervalo *para* $L$ *con* variable restringida $x$. *Si* $r$ *es un rango simple (resp. rango no-simple), entonces* $x \sqsubseteq r$ *es una* restricción de intervalo simple *(resp.* restricción de intervalo no-simple*). Si* $r$ *es consistente (resp. inconsistente),*

Figura 3.2: Estructura del dominio de intervalos simple $R_L^s$ donde $a, b, c \in L$ y $a \prec_L c \prec_L b$

*entonces* $x \sqsubseteq r$ *es* consistente *(resp.* inconsistente*). Si* $r = \top_{R_L^b}$*, entonces* $x \sqsubseteq r$ *es una* restricción de tipo para $x$*, y se denota como* $x ::' L$*. Si* $t \in L$*, entonces* $x = t$ *es equivalente a la restricción* $x \sqsubseteq [\mathbf{t}, \mathbf{t}]$*. El* dominio de las restricciones de intervalo sobre $X$ para $L$ *es el conjunto de todas las restricciones de intervalo para* $L$ *con variables restringidas en* $X$ *y se denota por* $\mathcal{C}_L^X$*. La union*

$$\mathcal{C}^X \stackrel{\text{def}}{=} \bigcup \{\mathcal{C}_L^X \mid L \in \mathcal{L}\}$$

*constituye el* dominio de la restricciones de intervalo sobre $X$ *para* $\mathcal{L}$*.*

*La relación de orden para* $\mathcal{C}^X$ *se hereda de la relación de orden definida en los dominios de intervalos. Nosotros definimos* $c_1 \preceq_{\mathcal{C}^X} c_2$ *si y sólo si, para algún* $L \in \mathcal{L}$*,* $c_1 = x \sqsubseteq r_1, c_2 = x \sqsubseteq r_2 \in \mathcal{C}_L^X$ *y* $r_1 \preceq_{R_L^b} r_2$*.*

**EJEMPLO 3.22** *Ejemplos de restricciones de tipo, de restricciones de intervalo simple y de restricciones de intervalo no-simples son mostradas a continuación en (a), (b) y (c), respectivamente, donde + (para* $L = \Re$*), − (para* $L = Integer$*) y trunc son tal como se definieron en los Ejemplos 3.9 y 3.11.*

$$\begin{array}{ll}
(a) & x, y ::' Integer, \qquad b ::' Bool, \qquad w, r ::' \Re; \\
(b) & y \sqsubseteq [\mathbf{1}, \mathbf{4}), \qquad b \sqsubseteq [true, true], \qquad r \sqsubseteq [\mathbf{1}, \mathbf{23}, \mathbf{3}, \mathbf{45}); \\
(c) & r \sqsubseteq min(w), max(w) + \mathbf{3}, \mathbf{2}] \quad y \\
 & x \sqsubseteq \overline{trunc}(min(w)) \overline{-} max(y), trunc(max(w)) - min(y).
\end{array}$$

**DEFINICIÓN 3.23** *(Intersección de restricciones de intervalo simple) Supón que* $x \in X$*. La intersección en un dominio* $L \in \mathcal{L}$ *de dos restricciones de intervalo simple* $c_1, c_2 \in \mathcal{C}_L^X$ *donde* $c_1 = x \sqsubseteq r_1$*,* $c_2 = x \sqsubseteq r_2$ *y* $x \in V_L$ *se define como sigue:*

$$c_1 \ \cap_L \ c_2 = glb_{\mathcal{C}_L^X}(c_1, c_2) = \ x \sqsubseteq glb_{R_L^s}(r_1, r_2).$$

Supón que $x \in X$ y $c_1, c_2, c_3 \in \mathcal{C}_L^X$ son restricciones de intervalo simple con variable restringida $x$ donde $c_3 = c_1 \cap_L c_2$. Entonces se deduce, a partir de la definición mostrada previamente, que $\cap_L$ posee las siguientes propiedades:

*Convergencia*: $c_3 \preceq_{\mathcal{C}_L^X} c_1$ y $c_3 \preceq_{\mathcal{C}_L^X} c_2$;

*Corrección*: si $c \preceq_{\mathcal{C}_L^X} c_1$ y $c \preceq_{\mathcal{C}_L^X} c_2$, entonces $c \preceq_{\mathcal{C}_L^X} c_3$;

*Conmutatividad*: $(c_1 \ \cap_L \ c_2) = (c_2 \ \cap_L \ c_1)$;

*Idempotencia*: $(c_1 \ \cap_L \ c_3) = c_3$ y $(c_3 \ \cap_L \ c_2) = c_3$.

Si $S_x$ es un conjunto de restricciones de intervalo simple restringidas sobre la misma variable $x \in V_L$ y sobre algún $L \in \mathcal{L}$, entonces nosotros definimos $\bigcap_L S_x = glb_{\mathcal{C}_L^X}(S_x)$. Como consecuencia de la propiedad de convergencia, $\bigcap_L S_x \preceq c$, para cada $c \in S_x$.

**EJEMPLO 3.24** *Considera los dominios* $Integer, \Re, Bool$ *y* $Set\ Integer$. *Sea* $X = \{i, r, b, s\}$ *donde* $i \in V_{Integer}$, $r \in V_{\Re}$, $b \in V_{Bool}$ *y* $s \in V_{Set\ Integer}$. *Entonces*

$$\begin{aligned} i \sqsubseteq [\mathbf{5}, \mathbf{24}] \ \cap_L \ i \sqsubseteq [\mathbf{1}, \mathbf{15}) &= glb_{\mathcal{C}^X_{Integer}}\big(i \sqsubseteq [\mathbf{5}, \mathbf{24}], i \sqsubseteq [\mathbf{1}, \mathbf{15})\big) \\ &= i \sqsubseteq glb_{R^b_{Integer}}\big([\mathbf{5}, \mathbf{24}], [\mathbf{1}, \mathbf{15})\big) \\ &= i \sqsubseteq glb_{\overline{Integer^b}}\big([\mathbf{5}, [\mathbf{1}), glb_{Integer^b}(\mathbf{24}], \mathbf{15})\big) \\ &= i \sqsubseteq [\mathbf{5}, \mathbf{15}). \end{aligned}$$

*También,*

$$\begin{aligned} r \sqsubseteq [\mathbf{1},\mathbf{12}, \mathbf{5},\mathbf{67}) \ \cap_L \ r \sqsubseteq [\mathbf{2},\mathbf{34}, \mathbf{5},\mathbf{95}) &= r \sqsubseteq [\mathbf{2},\mathbf{34}, \mathbf{5},\mathbf{67}); \\ b \sqsubseteq (\mathit{false},\mathit{true}] \ \cap_L \ b \sqsubseteq [\mathit{false},\mathit{true}] &= b \sqsubseteq (\mathit{false},\mathit{true}]; \\ s \sqsubseteq [\{\mathbf{1}\}, \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}] \ \cap_L \ s \sqsubseteq [\{\mathbf{3}\}, \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}\}] &= s \sqsubseteq [\{\mathbf{1}, \mathbf{3}\}, \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}]. \end{aligned}$$

### 3.4.2. Reducción de Restricciones

La reducción de las restricciones se llama estabilización de las restricciones y el proceso se basa principalmente en la intersección de restricciones de intervalo simple.

**DEFINICIÓN 3.25** *(Almacén de restricciones) Un* almacén de restricciones $S$ para $X$ *es un subconjunto finito de* $\mathcal{C}^X$. *El almacén* $S$ *es* estable *si, para cada* $x \in X$, *hay exactamente una única restricción de intervalo simple* $c$ *para* $x$ *en* $S$. *El almacén* $S$ *es* simple *si contiene sólo restricciones simples. El almacén* $S$ *es* inconsistente *si contiene al menos una restricción de intervalo inconsistente; en otro caso es* consistente.

*El* dominio de los almacenes simples de restricciones para $X$ *es el conjunto de todos los almacenes simples de restricciones para* $X$ *y lo denotamos por* $\mathcal{S}^X$. *El* dominio de los almacenes simples y estables de restricciones para $X$ *es el conjunto de todos los almacenes simples y estables de restricciones para* $X$ *y lo denotamos por* $\mathcal{SS}^X$.

**DEFINICIÓN 3.26** *(Almacén estabilizado) Sea* $S \in \mathcal{S}^X$ *y, para cada* $x \in X$, $S_x$ *el conjunto de restricciones en* $S$ *con variable restringida* $x$. *Entonces, el* almacén estabilizado $S' \in \mathcal{SS}^X$ *de* $S$ *se define como:*

$$S' = \{\cap_L(S_x) \mid x \in X \wedge x \in V_L \ \textit{(para algún } L \in \mathcal{L}) \ \}$$

*Escribimos* $S \mapsto S'$ *para expresar que* $S'$ *es el almacén estabilizado de* $S$.

Observa que, por la Definición 3.21, si $S_x = \emptyset$ entonces $\cap_L(S_x) = x \sqsubseteq \top_{R^b_L}$. Esto asegura que $S' \in \mathcal{SS}^X$.

**EJEMPLO 3.27** *Supón que* $Integer, \Re \in \mathcal{L}$ *y* $X = \{x, y, i\}$ *donde* $x, y \in V_{\Re}$ *y* $i \in V_{Integer}$. *Considera*

$$\begin{aligned} S = \big\{ \ & x \sqsubseteq (\mathbf{8},\mathbf{3}, \mathbf{20},\mathbf{4}], \quad y \sqsubseteq [\mathbf{1},\mathbf{2}, \mathbf{10},\mathbf{5}), \quad i \sqsubseteq [\mathbf{0}, \mathbf{10}], \\ & x \sqsubseteq [\mathbf{1},\mathbf{0}, \mathbf{15},\mathbf{0}], \quad y \sqsubseteq (\mathbf{5},\mathbf{6}, \mathbf{15},\mathbf{3}), \quad i \sqsubseteq [\mathbf{2}, \mathbf{15}) \ \big\}, \\ S' = \big\{ \ & x \sqsubseteq (\mathbf{8},\mathbf{3}, \mathbf{15},\mathbf{0}], \quad y \sqsubseteq (\mathbf{5},\mathbf{6}, \mathbf{10},\mathbf{5}), \quad i \sqsubseteq [\mathbf{2}, \mathbf{10}] \ \big\}. \end{aligned}$$

*Entonces* $S \mapsto S'$.

**DEFINICIÓN 3.28** *(Un orden parcial sobre los almacenes de restricciones simples y estables) Sean $S, S' \in \mathcal{SS}^X$ donde $c_x, c'_x$ denotan las restricciones de intervalo (simples) para $x \in X$ en $S$ y $S'$, respectivamente. Entonces $S \preceq_s S'$ si y sólo si, para cada $x \in X$, $c_x \preceq_{\mathcal{C}^X} c'_x$.*

Por tanto, $\mathcal{SS}^X$ forma un retículo con la relación de orden $\preceq_s$, cuyo máximo elemento (i.e., el top) es el conjunto de restricciones de tipo para $X$ (i.e., el conjunto $\{x \sqsubseteq \top_{R^s_L} \mid x \in X \cap V_L, L \in \mathcal{L}\}$) y cuyo mínimo elemento (i.e., el bottom) es el conjunto $\{x \sqsubseteq \bot_{R^s_L} \mid x \in X \cap V_L, L \in \mathcal{L}\}$.

**EJEMPLO 3.29** *Considera los dominios Integer y Bool. Entonces*

$$\big\{b \sqsubseteq [\mathit{false},\mathit{false}], i \sqsubseteq (\mathbf{2},\mathbf{4}]\big\} \prec_s \big\{b \sqsubseteq [\mathit{false},\mathit{true}], i \sqsubseteq [\mathbf{1},\mathbf{5})\big\}.$$

**PROPOSICIÓN 3.30** *Sean $X' \subseteq X$ y $C \in \mathcal{S}^{X'}$. Sean también $S, S' \in \mathcal{SS}^X$ tal que $S \cup C \mapsto S'$. Entonces, $S' \preceq_s S$ y $\forall c \in C : \exists c' \in S' \,.\, c' \preceq_{\mathcal{C}^X} c$.*

Cualquier almacén de restricciones (in)consistente en $\mathcal{SS}^X$ identifica un conjunto de almacenes de restricciones (in)consistentes en $\mathcal{SS}^X$.

**PROPOSICIÓN 3.31** *Supón $S, S' \in \mathcal{SS}^X$ donde $S \preceq_s S'$. Entonces, si $S'$ es inconsistente, $S$ es también inconsistente.*

### 3.4.3. Propagación de Restricciones

La propagación de restricciones se define mediante una función de evaluación.

**DEFINICIÓN 3.32** *(Evaluación de términos indexicals) Sean $S \in \mathcal{SS}^X$, $x \in X$, y*

$$\mathcal{L}^X = \cup\big\{L^X \,\big|\, L \in \mathcal{L}, L^X = \{t \in L^b \mid \mathit{vars}(t) \subseteq X\}\big\},$$
$$\overline{\mathcal{L}^X} = \cup\big\{\overline{L^X} \,\big|\, L \in \mathcal{L}, \overline{L^X} = \{t \in \overline{L^b} \mid \mathit{vars}(t) \subseteq X\}\big\}$$

*donde $\mathit{vars}(t)$ denota el conjunto de todas las variables ocurriendo en $t$. Entonces las funciones de evaluación (sobrecargadas) se definen como::*

$$\mathit{eval} :: \mathcal{SS}^X \times \mathcal{L}^X \to \mathcal{L}^X, \qquad \mathit{eval} :: \mathcal{SS}^X \times \overline{\mathcal{L}^X} \to \overline{\mathcal{L}^X},$$

$$\begin{aligned}
\mathit{eval}(S,t) &= t && \text{si } t \in L^s \cup \overline{L^s},\ L \in \mathcal{L},\\
\mathit{eval}(S,\mathit{max}(x)) &= t && \text{donde } x \sqsubseteq \langle \overline{s}, t\rangle \in S,\\
\mathit{eval}(S,\mathit{min}(x)) &= \overline{s} && \text{donde } x \sqsubseteq \langle \overline{s}, t\rangle \in S,\\
\mathit{eval}(S,\mathit{val}(x)) &= t && \text{si } x \sqsubseteq \langle \overline{t}, t\rangle \in S,\\
\mathit{eval}(S,\mathit{val}(x)) &= \mathit{val}(x) && \text{si } x \sqsubseteq \langle \overline{t}, t\rangle \notin S,\\
\mathit{eval}(S,\overline{\mathit{val}(x)}) &= \overline{t} && \text{si } x \sqsubseteq \langle \overline{t}, t\rangle \in S,\\
\mathit{eval}(S,\overline{\mathit{val}(x)}) &= \overline{\mathit{val}(x)} && \text{si } x \sqsubseteq \langle \overline{t}, t\rangle \notin S,\\
\mathit{eval}(S,\circ(t_1,\ldots,t_n)) &= \circ(\mathit{eval}(S,t_1),\ldots,\mathit{eval}(S,t_n)),\\
\mathit{eval}(S,\overline{\circ}(t_1,\ldots,t_n)) &= \overline{\circ}(\mathit{eval}(S,t_1),\ldots,\mathit{eval}(S,t_n)).
\end{aligned}$$

**EJEMPLO 3.33** *Sea*

$$S = \{x \sqsubseteq (\mathbf{1},\mathbf{23},\mathbf{6},\mathbf{78}], y \sqsubseteq [\mathbf{4},\mathbf{54},\mathbf{3},\mathbf{41}), i \sqsubseteq (\mathbf{1},\mathbf{4}], b \sqsubseteq [\mathit{false},\mathit{false}]\ \}$$

*un almacén estable de restricciones simples para* $\{x, y, i, b\}$ *i.e.,* $S \in \mathcal{SS}^{\{x,y,i,b\}}$*. Entonces*

$$\begin{aligned}
\mathit{eval}(S, \mathbf{2},\mathbf{34}]) &= \mathbf{2},\mathbf{34}],\\
\mathit{eval}(S, \mathit{min}(i)\overline{+}[\mathbf{3}) &= \mathit{eval}(S, \mathit{min}(i))\overline{+}\mathit{eval}(S, [\mathbf{3}) = (\mathbf{1}\overline{+}[\mathbf{3} = (\mathbf{4},\\
\mathit{eval}(S, \mathit{trunc}(\mathit{max}(x))) &= \mathit{trunc}(\mathit{eval}(S, \mathit{max}(x))) = \mathit{trunc}(\mathbf{6},\mathbf{78}]) = \mathbf{6}],\\
\mathit{eval}(S, \overline{\mathit{val}(b)}) &= [\mathit{false},\\
\mathit{eval}(S, \mathit{val}(x)) &= \mathit{val}(x).
\end{aligned}$$

*donde* $+/2$ *(para* $L = Integer$*) y* $trunc/1$ *se definen, respectivamente, igual que en los Ejemplos 3.9 y 3.11.*

Nota que nuestros términos indexicals son una generalización de los términos de indexicals de clp(FD) (Codognet y Diaz, 1996a) y que admiten rangos finitos e infinitos.

Sea $c = x \sqsubseteq \langle \overline{s}, t\rangle$ una restricción de intervalo en $\mathcal{C}_L^X$ y, por tanto, $\overline{s} \in \overline{L^b}$ y $t \in L^b$ para algún $L \in \mathcal{L}$. Entonces sobrecargamos *eval/2* y definimos $\mathit{eval}(S, c)$ como

$$\mathit{eval}(S, c) = x \sqsubseteq \big\langle \mathit{eval}(S, \overline{s}), \mathit{eval}(S, t)\big\rangle.$$

**DEFINICIÓN 3.34** *(Propagación de restricciones) Supón que* $S \in \mathcal{SS}^X$*.*

*Si* $c, c' \in \mathcal{C}_L^X$ *donde* $c'$ *es simple y* $\mathit{eval}(S, c) = c'$*, entonces decimos que* $c$ se propaga (usando $S$) *a* $c'$ *y escribimos* $c \rightsquigarrow^S c'$*.*

*Si* $C \subseteq \mathcal{C}^X$ *y* $C' = \{c' \mid \exists c \in C\ .\ c \rightsquigarrow^S c'\}$ *entonces decimos que* $C$ se propaga a $C'$ using $S$ *y escribimos* $C \rightsquigarrow^S C'$*. Como consecuencia,* $C' \subseteq \mathcal{C}^{X'}$ *es un almacén de restricciones simples donde* $X' \subseteq X$*.*

Observe que, si $x \sqsubseteq \langle \overline{s}, t\rangle \in S$ donde $s \neq t$, entonces la función de evaluación *eval* aplicada a $\mathit{val}(x)$ (resp. $\overline{\mathit{val}(x)}$) devuelve $\mathit{val}(x)$ (resp. $\overline{\mathit{val}(x)}$), es decir, equivale a la identidad. Por lo tanto el indexical $\mathit{val}(x)$ (resp. $\overline{\mathit{val}(x)}$) suministra un mecanismo útil para retrasar la propagación de las restricciones.

Nota también que, si $C$ es finito y $C \rightsquigarrow^S C'$, entonces $C'$ puede computarse (i.e., el proceso de propagar $C$ a $C'$ usando $S$ termina).

**EJEMPLO 3.35** *Considera el almacén* $S'$ *y las variables del Ejemplo 3.27, y el operador* $trunc/1$ *tal como fue definido en el Ejemplo 3.11. Entonces*

$$\Big\{x \sqsubseteq \mathit{min}(y),\mathbf{20},\mathbf{4}],\ i \sqsubseteq \overline{\mathit{trunc}}\big(\mathit{min}(y)\big),\mathit{trunc}\big(\mathit{max}(y)\big)\Big\} \rightsquigarrow^{S'} \big\{x \sqsubseteq (\mathbf{5},\mathbf{6},\mathbf{20},\mathbf{4}],\ i \sqsubseteq [\mathbf{5},\mathbf{10}]\big\}.$$

### 3.4.4. Equivalencia en el Dominio Discreto

Cuando el dominio de computación $L$ es discreto, podemos identificar elementos equivalentes en $L^s$ y en su reflejo $\overline{L^s}$ y, como consecuencia, en el dominio de intervalos $R_L^s$ y el dominio de las restricciones $\mathcal{C}_L^X$.

**EJEMPLO 3.36** *Para cualquier $i \in Integer$, el predecesor inmediato es $i-1$ de manera que, por ejemplo,* $\mathbf{3}) = \mathbf{2}]$. *Similarmente, con el dominio Bool, el predecesor inmediato de 'true' es 'false' de manera que* $\mathbf{true}) = \mathbf{false}]$.

Supón que $L$ es un dominio discreto en el cual el *predecesor inmediato* $pre$(a) de cada valor a $\in L$, con a $\neq \bot_L$, está definido. Para reducir el tamaño del dominio $L^s$, introducimos la siguiente *regla de equivalencia* para cualquier $\mathbf{a}) \in L^s$ y a $\neq \bot_L$

$$\mathbf{a}) \equiv pre(\mathrm{a})] \quad \text{en } L^s.$$

Por el principio de dualidad de los retículos, en el dominio $\overline{L^s}$ (donde $succ$(a) es el *sucesor inmediato* de cualquier a $\in L$ con a $\neq \top_L$), tenemos

$$(\mathbf{a} \equiv [succ(\mathrm{a}) \quad \text{en } \overline{L^s}.$$

Si los elementos $\bot_L$ y $\top_L$ fueron añadidos como cotas ficticias, entonces definimos

$$pre(\top_L) \equiv \top_L, \qquad succ(\bot_L) \equiv \bot_L.$$

Cuando el dominio de intervalos se construye a partir de un dominio discreto $L$, las reglas de equivalencia permiten reducir el tamaño de $L^s$ y $\overline{L^s}$ y por tanto, el tamaño de $R_L^s$. Si *pre/1* esta definido para cada elemento en $L$ (con la excepción, ya comentada, de $\bot_L$), entonces se puede proporcionar una forma canónica para $L^s$ de manera que los paréntesis abiertos '(','(' se eliminan y son reemplazados por los paréntesis cerrados '[','['.

**EJEMPLO 3.37** *Para el dominio Integer,* $[\mathbf{1},\mathbf{3}) \equiv [\mathbf{1},\mathbf{2}]$. *Similarmente, para el dominio Bool,* $[false, true) \equiv [false, false]$. *Supón que* $L = \{0,1,2,3\}$ *es un retículo donde* $0 < 1 < 2 < 3$. *Entonces,*

$$L^s = \{\ \mathbf{0}),\mathbf{0}],\mathbf{1}),\mathbf{1}],\mathbf{2}),\mathbf{2}],\mathbf{3}),\mathbf{3}]\ \} \equiv \{\ \mathbf{0}),\mathbf{0}],\mathbf{1}],\mathbf{2}],\mathbf{3}]\ \},$$
$$\overline{L^s} = \{\ (\mathbf{3},[\mathbf{3},(\mathbf{2},[\mathbf{2},(\mathbf{1},[\mathbf{1},(\mathbf{0},[\mathbf{0}\ \} \equiv \{\ (\mathbf{3},[\mathbf{3},[\mathbf{2},[\mathbf{1},[\mathbf{0}\ \}.$$

*Supón también que* $X = \{x, b\}$, $x \in V_{Integer}$ *y* $b \in V_{Bool}$. *Entonces,* $x \sqsubseteq (\mathbf{1},\mathbf{5})$ *es equivalente a* $x \sqsubseteq [\mathbf{2},\mathbf{4}]$ *en* $\mathcal{C}^X_{Integer}$ *y* $b \sqsubseteq (false, true]$ *es equivalente a* $b \sqsubseteq [true, true]$ *en* $\mathcal{C}^X_{Bool}$.

Con estas reglas de equivalencia para los dominios discretos, más inconsistencias pueden ser detectadas.

**EJEMPLO 3.38** *Considera de nuevo el dominio $L$ del Ejemplo 3.20. Supón que $L$ es discreto y que $pre(c) = \mathbf{a}$ y $pre(b) = \mathbf{c}$. Entonces los rangos $(\mathbf{a}, \mathbf{c}) \in R^s_L$ y $(\mathbf{c}, \mathbf{b})$ son inconsistentes ya que equivalen a los rangos $[\mathbf{c}, \mathbf{a}]$ y $[\mathbf{b}, \mathbf{c}]$ respectivamente. Estos son los nodos marcados con círculos en la Figura 3.2.*

*Por ejemplo, sea $L = Integer$. Entonces el rango $(\mathbf{1}, \mathbf{2})$ es inconsistente en el dominio $R^s_{Integer}$ pues $[\mathbf{2}, \mathbf{1}]$ es inconsistente. También la restricción $x \sqsubseteq (\mathbf{1}, \mathbf{2})$ es inconsistente pues $x \sqsubseteq [\mathbf{2}, \mathbf{1}]$ es inconsistente.*

En el resto del documento, asumimos que todas las restricciones de intervalos simples definidas sobre dominios discretos son reducidas (siempre que sea posible) mediante la equivalencia mostrada en esta sección.

### 3.4.5. Una Solución para un Almacén de Restricciones

**DEFINICIÓN 3.39** *(Solución) Sea $C \in \wp(\mathcal{C}^X)$ un almacén de restricciones para $X$. Una* solución para $C$ *es un almacén de restricciones $R \in \mathcal{SS}^X$ donde,*

$$C \rightsquigarrow^R C'$$
$$R \cup C' \mapsto R.$$

*El conjunto de todas las soluciones para $C$ lo denotamos como $Sol(C)$. Si existe, la* solución más general para $C$ *(*m.g.s. para $C$*), se define como:*

$$m.g.s.\ para\ C = G \in Sol(C) \mid \forall R \in Sol(C).R \preceq_s G.$$

Por tanto, una solución es un almacén de restricciones consistente que contiene, para cada una de las variables restringidas, exactamente una única restricción de intervalo simple que no puede ser reducida mediante la propagación de las restricciones.

**LEMA 3.40** *Sean $C \in \wp(\mathcal{C}^X)$ y $S, R \in \mathcal{SS}^X$. Si $R$ es una solución para $C \cup S$, entonces, $R \preceq_s S$.*

### 3.4.6. Monotonía de las Restricciones

En esta sección mostramos que todas las restricciones que pueden ser definidas en nuestro marco teórico son forzosamente monótonas.

**LEMA 3.41** *Supón que $c \preceq_{\mathcal{C}^X} c'$ y que $c$ y $c'$ son restricciones consistentes simples para $L \in \mathcal{L}$ restringidas sobre la misma variable $y \in X$ y supón también que $c' = y \sqsubseteq \langle \overline{t'}, t' \rangle$ para algún $t' \in L^s$. Entonces $c = c'$.*

Cada restricción de intervalo se propaga de forma monótona con respecto a la relación de orden definida sobre los almacenes estables simples.

**LEMA 3.42** *Sean $S_1, S_2 \in \mathcal{SS}^X$ dos almacenes estables y consistentes tal que $S_1 \preceq_s S_2$ y $c, c_2 \in \mathcal{C}^X$ de manera que $c \rightsquigarrow^{S_2} c_2$. Entonces, existe $c_1 \in \mathcal{C}^X$ tal que $c \rightsquigarrow^{S_1} c_1$ y $c_1 \preceq_{\mathcal{C}^X} c_2$.*

**PROPOSICIÓN 3.43** *Sean $S_1, S_2 \in \mathcal{SS}^X$ donde $S_1$ y $S_2$ son consistentes, $S_1 \preceq_s S_2$ y $C \in \wp(\mathcal{C}^X)$ de manera que*

$$C \rightsquigarrow^{S_1} C_1 \text{ y } S_1 \cup C_1 \mapsto S_1',$$
$$C \rightsquigarrow^{S_2} C_2 \text{ y } S_2 \cup C_2 \mapsto S_2'.$$

*Entonces $S_1' \preceq_s S_2'$.*

**EJEMPLO 3.44** *Considera la definición del operador $-$ del Ejemplo 3.9, cuando $L$ es $\Re$ así que $- :: \Re^s \times \overline{\Re^s} \to \Re^s$. Supón que $X = \{x, y\}$, $x, y \in V_\Re$,*

$$S_1 = \{\ y \sqsubseteq (\mathbf{2},\mathbf{0},\mathbf{4},\mathbf{0}],\ x ::' \Re\ \} \quad y \quad S_2 = \{\ y \sqsubseteq (\mathbf{1},\mathbf{0},\mathbf{11},\mathbf{0}],\ x ::' \Re\ \}.$$

*Entonces $S_1, S_2 \in \mathcal{SS}^X$ y $S_1 \prec_s S_2$. Supón que*

$$c_1 = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{20},\mathbf{0}]\text{–}min(y) \quad y \quad c_2 = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{20},\mathbf{0}]\text{–}max(y).$$

*Entonces $c_1$ es una restricción de intervalo para $\Re$ porque*

$$[\mathbf{0},\mathbf{0} \in \overline{\Re^b} \text{ y } \mathbf{20},\mathbf{0}]\text{–}min(y) \in \Re^b$$

*y, por tanto,*

$$[\mathbf{0},\mathbf{0},\mathbf{20},\mathbf{0}]\text{–}min(y) \in R^b_\Re.$$

*Usando el procedimiento de propagación de restricciones para $S_1$ y $S_2$ obtenemos*

$$c_1 \rightsquigarrow^{S_1} c_{11}, \qquad c_1 \rightsquigarrow^{S_2} c_{12}$$

*donde*

$$c_{11} = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{18},\mathbf{0}), \qquad c_{12} = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{19},\mathbf{0}).$$

*Entonces, tenemos $c_{11} \prec_{\mathcal{C}^X} c_{12}$ (i.e., $c_1$ se propagó monótonamente con respecto a la relación de orden definida entre los almacenes de restricción estables).*

*Sin embargo, $c_2$ no es una restricción de intervalo. Esto se debe a que, aunque $[\mathbf{0},\mathbf{0} \in \overline{\Re^b} \text{ y } \mathbf{20},\mathbf{0}] \in \Re^b$. Vemos que $max(y) \notin \overline{\Re^b}$ con la consecuencia de que,*

$$\mathbf{20},\mathbf{0}]\text{–}max(y) \notin \overline{\Re^b}$$

*y, por tanto,*

$$[\mathbf{0},\mathbf{0},\mathbf{20},\mathbf{0}]\text{–}max(y) \notin R^b_\Re.$$

*Observa que si aplicásemos el procedimiento de propagar esta restricción, obtendríamos*

$$c_2 \rightsquigarrow^{S_1} c_{21}, \qquad c_2 \rightsquigarrow^{S_2} c_{22}$$

*donde*

$$c_{21} = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{16},\mathbf{0}], \qquad c_{22} = x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{9},\mathbf{0}].$$

*Con lo cual* $c_{22} \prec_{\mathcal{C}^X} c_{21}$. *Por lo tanto, usando al almacén de restricciones más pequeño* $S_1$, $c_2$ *da lugar a un rango más grande para* $x$. *El problema es causado por el hecho de que si* $S_2$ *se reemplaza por un almacén menor, tal como* $S_1$, $max(y)$ *también disminuye en* $\Re^s$, *así que el valor de* **20**,**0**]–$max(y)$ *aumenta. Por tanto, la cota derecha del rango para* $x$ *en* $c_2$ *también incrementa de manera que el límite superior para* $y$ *nunca podría ser reducido.*

*Afortunadamente este problema se detecta simplemente asegurando la validez de las restricciones en nuestro marco teórico. Observa que la aceptabilidad de las restricciones, tales como* $c_2$, *puede decidirse* a priori *mediante el uso de técnicas estándares de comprobación de tipos.*

## 3.5. Semántica Operacional

En esta sección, proporcionamos un esquema operacional para la propagación de las restricciones de intervalo y probamos la corrección y la terminación de este esquema.

### 3.5.1. Esquema Operacional para la Propagación de Restricciones

Sea $C \in \wp_f(\mathcal{C}^X)$ y $S \in \mathcal{SS}^X$. Definimos aquí *propaga*$(C, S)$, un *esquema operacional* para calcular una solución (si existe) para $C \cup S$.

*procedimiento propaga*$(C, S)$
*empieza*
  *si* $S$ *is consistente entonces* (0)
    $C := C \cup S$; (1)
    *repite*
      $C \rightsquigarrow^S C'$; %% Propagación de Restricciones (2)
      $S' := S$; (3)
      $S' \cup C' \mapsto S$; %% Estabilización (4)
    *hasta que* $S$ *sea inconsistente o* $S = S'$; (5)
  *finsi;*
*finempieza.*

Figura 3.3: *propaga*/2: un esquema genérico para la propagación de las restricciones de intervalo

Si al menos una solución existe, el almacén $S$ contiene una solución para $C \cup S$.

**TEOREMA 3.45** *(Corrección) Sean* $C \in \wp_f(\mathcal{C}^X)$ *y* $S \in \mathcal{SS}^X$. *Si al menos una solución para* $C \cup S$ *existe, entonces cualquier ejecución del esquema operacional para*

*propaga*$(C, S)$ *que termine, devuelve en* $S$ *la solución más general para* $C \cup S$*. En otro caso, si no existe ninguna solución para* $C \cup S$*, entonces a la finalización de* *propaga*$(C, S)$*,* $S$ *es inconsistente.*

### 3.5.2. Terminación

El paso de propagación (2), en la Figura 3.3, añade nuevas restricciones creadas antes del paso de estabilización (4). Por tanto, con dominios infinitos, el procedimiento podría no terminar (observa que, en dominios de computación infinitos, las restricciones podrían ser contraídas indefinidamente mediante el paso de estabilización).

**EJEMPLO 3.46** *Considera el operador* $div2_{\Re} :: \Re \to \Re$ *donde* $div2_{\Re}\ \mathbf{a} = \frac{\mathbf{a}}{2,0}$ *para cualquier* $a \in \Re$*, y sea* $div2_B$ *definido como la identidad sobre* $B$ *(i.e., el dominio de los paréntesis). Sea también,* $C$ *el almacén de restricciones*

$$\begin{array}{l}\{\ x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{10},\mathbf{0}],\ x \sqsubseteq [\mathbf{0},\mathbf{0},\mathit{div2}(\mathit{max}(y)),\\ \quad y \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{10},\mathbf{0}],\ y \sqsubseteq [\mathbf{0},\mathbf{0},\mathit{div2}(\mathit{max}(x))\ \}\end{array}$$

*donde* $x, y \in V_{\Re}$ *y* $S_0$ *es el elemento máximo del retículo* $\mathcal{SS}^{\{x,y\}}$*. Sea también* $S_i$ *el valor del almacén* $S$ *al final de la* $i$ *iteración, para* $i \geq 1$*, del esquema operacional para* *propaga*$(C, S)$ *siendo* $S_0$ *el valor inicial de* $S$*. Entonces, durante la ejecución de* *propaga*$(C, S)$*,* $S$ *se reduce indefinidamente i.e.,*

$$\begin{array}{l}S_0 = \{\ x \sqsubseteq [\bot_{\Re},\top_{\Re}], y \sqsubseteq [\bot_{\Re},\top_{\Re}]\ \},\\ S_1 = \{\ x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{10},\mathbf{0}],\ y \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{10},\mathbf{0}]\ \},\\ S_2 = \{\ x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{5},\mathbf{0}],\ y \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{5},\mathbf{0}]\ \},\\ S_3 = \{\ x \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{2},\mathbf{5}],\ y \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{2},\mathbf{5}]\ \},\\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots\end{array}$$

Para forzar la terminación, introducimos la noción de *precision de una restricción consistente.*

**DEFINICIÓN 3.47** *(Precisión de una restricción) Sea* $\mathcal{CC}_L^X$ *el conjunto de todas las restricciones de intervalo consistentes para* $L$ *y con variables restringidas en* $X$*,* $x \in X \cap V_L$ *para cualquier* $L \in \mathcal{L}$ *y sea* $\Re\mathcal{I}$ *el producto lexicográfico* $(\Re^+, Integer)$ *donde* $\Re^+$ *es el dominio de los números reales no-negativos. Entonces definimos*

$$\begin{array}{l}precision_L :: \mathcal{CC}_L^X \to \Re\mathcal{I}\\ precision_L(\ x \sqsubseteq \{\!\}_1 a,b \{\!\}_2\ ) = (\ \hat{a} \diamond_L b,\ \{\!\}_1 \diamond_B \{\!\}_2\ )\end{array} \tag{3.3}$$

*donde* $\diamond_L :: \{\ (\hat{a}, b)\ \mid\ a, b \in L, a \preceq_L b\ \} \to \Re^+$ *es una función monótona estricta (bien predefinida o bien definida por el usuario) y* $\diamond_B :: B \times B \to \{0, 1, 2\}$ *es la función monótona estricta*

$$\text{`]'} \diamond_B \text{`]'} \stackrel{\text{def}}{=} 2 \qquad \text{`]'} \diamond_B \text{`)'} \stackrel{\text{def}}{=} 1 \qquad \text{`)'} \diamond_B \text{`]'} \stackrel{\text{def}}{=} 1 \qquad \text{`)'} \diamond_B \text{`)'} \stackrel{\text{def}}{=} 0.$$

Puesto que $precision_L$ se define sólo sobre restricciones consistentes, la función $\diamond_L$ sólo tiene que ser definida sobre aquellos rangos en los que su primer argumento (i.e., la cota inferior) es menor o igual que su segundo argumento (i.e., la cota superior). Esta función tiene que ser definida para todos los dominios de computación.

**PROPOSICIÓN 3.48** *$precision_L$ es estrictamente monótona. i.e.,*

$$precision_L(c) <_{\Re\mathcal{I}} precision_L(c') \text{ si } c <_{\mathcal{CC}_L^X} c'.$$

**EJEMPLO 3.49** *Sean Integer, $\Re$, Bool y Set Integer y también $\Re^2 = \langle \Re, \Re \rangle$. Entonces definimos, para todo $i_1, i_2 \in Integer$, $x_1, x_2, y_1, y_2 \in \Re$, $b_1, b_2 \in Bool$ y $s_1, s_2 \in Set\ Integer$ tal que $i_1 \preceq_{Integer} i_2$, $x_1 \preceq_{\Re} x_2$, $y_1 \preceq_{\Re} y_2$, $b_1 \preceq_{Bool} b_2$ y $s_1 \preceq_{Set\ Integer} s_2$,*

$$\begin{aligned}
\hat{i_1} \diamond_{Integer} i_2 &= i_2 - i_1, \\
\hat{x_1} \diamond_{\Re} x_2 &= x_2 - x_1, \\
\hat{b_1} \diamond_{Bool} b_2 &= 0{,}0 \text{ si } b_1 = b_2 \text{ y } 1{,}0 \text{ en otro caso}, \\
\widehat{\langle x_1, y_1 \rangle} \diamond_{\Re^2} \langle x_2, y_2 \rangle &= +\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \\
\hat{s_1} \diamond_{Set\ Integer} s_2 &= \#(s_2) - \#(s_1).
\end{aligned}$$

*Sea $X = \{i, i', x, y, s, b\}$ y supón que $i, i' \in V_{Integer}$, $x \in V_{\Re}$, $y \in V_{\Re^2}$, $s \in V_{Set\ Integer}$, $b \in V_{Bool}$ y*

$$\begin{aligned}
c_1 = i \sqsubseteq [\mathbf{1}, \mathbf{4}], \qquad & c_2 = i' \sqsubseteq [\mathbf{1}, \mathbf{4}), \\
c_3 = x \sqsubseteq (\mathbf{3}{,}\mathbf{5}, \mathbf{5}{,}\mathbf{7}), \qquad & c_4 = y \sqsubseteq [(\mathbf{2}{,}\mathbf{0}, \mathbf{3}{,}\mathbf{0}), (\mathbf{3}{,}\mathbf{4}, \mathbf{5}{,}\mathbf{6})], \\
c_5 = s \sqsubseteq [\{\}, \{\mathbf{3}, \mathbf{4}, \mathbf{5}\}) \quad & y\ c_6 = b \sqsubseteq [false, true].
\end{aligned}$$

*Entonces*

$$\begin{aligned}
&precision_{Integer}(c_1) = (3{,}0, 2), \qquad && precision_{Integer}(c_2) = (2{,}0, 2), \\
&precision_{\Re}(c_3) = (2{,}2, 0), && precision_{\Re^2}(c_4) = (2{,}95, 2), \\
&precision_{Set\ Integer}(c_5) = (3{,}0, 1), && precision_{Bool}(c_6) = (1{,}0, 2)
\end{aligned}$$

*Observa que $precision_{Integer}(c_2)$ es $(2{,}0, 2)$ en vez de $(3{,}0, 1)$ puesto que la restricción $i' \sqsubseteq [\mathbf{1}, \mathbf{4})$ es equivalente en el dominio $\mathcal{C}^X$ (ver la Sección 3.4.4) a la restricción $i' \sqsubseteq [\mathbf{1}, \mathbf{3}]$, y la precisión se calcula entonces con respecto a esta última restricción.*

*Los operadores usados en el ejemplo, es decir, $-$, $+$, $\#$ y la potencia al cuadrado tienen que estar completamente definidos sobre el dominio de computación y, por tanto, deben incluir las cotas liftadas (i.e., ficticias). A su vez, el operador raíz-cuadrada, definido sobre el dominio de los reales no-negativos, tiene que definirse también para la cota liftada superior.*

**DEFINICIÓN 3.50** *(Precisión de un almacén de restricciones) La precisión de un almacén de restricciones consistente $S \in \mathcal{SS}^X$ se define como*

$$precision(S) = \sum_{c \in S, c \in \mathcal{CC}_L^X \text{ para algún } L \in \mathcal{L}} precision_L(c)$$

*donde la suma en $\Re\mathcal{I}$ está definida como $(a_1, a_2) + (b_1, b_2) = (a_1 + b_1, a_2 + b_2)$.*

Por tanto, la precisión de un almacén de restricciones consistente y estable $S$ es igual a la suma de las precisiones de cada una de las restricciones que lo componen.

**EJEMPLO 3.51** *Supón que* $S = \{c_1, c_2, c_3, c_4, c_5, c_6\} \in \mathcal{SS}^X$, *donde, para* $1 \leq i \leq 6$, $c_i$ *se define como en el Ejemplo 3.49, y* $X = \{i, i', x, y, s, b\}$. *Entonces*

$$precision(S) = \sum_{1 \leq i \leq 5} precision_L(c_i) = (14{,}15, 9).$$

**PROPOSICIÓN 3.52** *Supón que* $S, S' \in \mathcal{SS}^X$ *son almacenes consistentes y que* $S \prec_s S'$. *Entonces* $precision(S) <_{\Re\mathcal{I}} precision(S')$.

Mediante la definición de una cota computable[2] $\varepsilon \in \Re^+ \cup \{0{,}0\}$, podemos comprobar si la precisión de las restricciones simples en un almacén $S$ fue reducida significativamente en el proceso de estabilización (step 4). Si el cambio es considerado suficientemente grande, entonces continuamos con el proceso de propagación. En otro caso, el conjunto de restricciones en $S$ se considera como "una solución bastante aceptable" y el procedimiento termina. Esta "solución" es una aproximación al concepto de solución mostrado en la Definición 3.39.

La cota $\varepsilon$ puede ser predefinida o definida por el usuario. Para usar $precision/1$ y $\varepsilon$, a continuación definimos un nuevo esquema operacional para $propaga_\varepsilon(C, S)$ que es el mismo que el esquema para $propaga(C, S)$ pero donde el paso (5) ha sido reemplazado por

($5^\star$) *hasta que* $S$ *sea inconsistente* o $precision(S') - precision(S) \leq_{\Re\mathcal{I}} (\varepsilon, 0)$.

**TEOREMA 3.53** *(Terminación) Sean* $C \in \wp_f(\mathcal{C}^X)$ *y* $S \in \mathcal{SS}^X$. *Si* $\varepsilon > 0{,}0$ *entonces el esquema operacional para* $propaga_\varepsilon(C, S)$ *termina.*

**DEFINICIÓN 3.54** *(Solución aproximada) Sean* $C \in \wp_f(\mathcal{C}^X)$ *y* $S \in \mathcal{SS}^X$. *Sea también* $R$ *una solución para* $C \cup S$ *y* $(\delta, \phi) \in \Re\mathcal{I}$. *Una* solución aproximada vía $(\delta, \phi)$ para $C \cup S$ *es un almacén estable de restricciones simples para* $X$, *denotado por* $R_{(\delta,\phi)}$, *donde,*

$$precision(R_{(\delta,\phi)}) - precision(R) \leq_{\Re\mathcal{I}} (\delta, \phi) \quad y \quad R \preceq_s R_{(\delta,\phi)}.$$

El número de iteraciones del esquema operacional depende, en los dominios infinitos, del valor de $\varepsilon$. En estos casos, la solución final para $propaga_\varepsilon(C, S)$ es una solución aproximada para $C \cup S$.

**TEOREMA 3.55** *(Corrección extendida) Sean* $C \in \wp_f(\mathcal{C}^X)$ *y* $S \in \mathcal{SS}^X$. *Si al menos una solución* $R$ *para* $C \cup S$ *existe, cualquier ejecución de* $propaga_\varepsilon(C, S)$ *que termine, devuelve en* $S$ *una solución aproximada para* $C \cup S$.

[2]Es decir, que puede ser representada en el ordenador que está siendo usado como máquina de computación.

También en estos casos, la solución aproximada depende del valor de $\varepsilon$ en el sentido de que, cuanto más pequeño sea $\varepsilon$, más próxima a la solución estará la solución aproximada.

**TEOREMA 3.56** *Sea $R$ una solución para $C \cup S$ donde $C \in \wp_f(\mathcal{C}^X)$ y $S \in \mathcal{SS}^X$. Supón que $S_{\varepsilon_1}$ y $S_{\varepsilon_2}$ son las soluciones aproximadas devueltas por el esquema operacional para propaga$_{\varepsilon_1}(C,S)$ y propaga$_{\varepsilon_2}(C,S)$, respectivamente. Entonces, si $0{,}0 \leq \varepsilon_1 \leq \varepsilon_2$,*

$$R \preceq_s S_{\varepsilon_1} \preceq_s S_{\varepsilon_2}.$$

La función *precision*/1 y la cota $\varepsilon$ permiten un control transparente y directo sobre la precisión de los resultados. Por ejemplo, podríamos definir $\varepsilon = 10^{-8}$ para los reales.

La terminación de nuestro algoritmo consiste en medir la propagación midiendo la diferencia entre las precisións de los almacenes de restricciones. Este concepto es genérico y, a nuestro entender, novedoso. Un enfoque más básico, pero sólo sobre los reales, consiste en asociar un parámetro que mida la precisión para limitar el número de veces que una variable puede ser contraída (Sidebottom y Havens, 1992).

## 3.6. Instancias

### 3.6.1. Dominios Clásicos

La mayoría de los dominios de computación tradicionales para la propagación de restricciones son retículos. Por ejemplo, *Integer*, $\Re$, *Bool*, y *Set* $L$ (para cualquier dominio $L \in \mathcal{L}$). En estos dominios es posible reducir restricciones de intervalo de una forma natural, tal y como se muestra en los siguientes ejemplos:

(1) $i \sqsubseteq \mathbf{[1,8)} \ \cap_L \ i \sqsubseteq \mathbf{(0,5]} \ = \ i \sqsubseteq \mathbf{[1,5]}$;
(2) $r \sqsubseteq \mathbf{[1.12,5.67)} \ \cap_L \ r \sqsubseteq \mathbf{[2.34,5.95)} \ = \ r \sqsubseteq \mathbf{[2.34,5.67)}$;
(3) $b \sqsubseteq (\mathit{false}, \mathit{true}] \ \cap_L \ b \sqsubseteq [\mathit{false}, \mathit{true}] \ = \ b \sqsubseteq (\mathit{false}, \mathit{true}]$.

Supón también que $L_1 \in \mathcal{L}$. Entonces, *Set* $L_1$ es un retículo sobre el cual es posible reducir las restricciones. Por ejemplo, considera el siguiente almacén de restricciones:

$$\begin{aligned} S \ = \ \{ \ & s ::' \mathit{Set\ Integer}, \\ & s \sqsubseteq \mathbf{[\{1\},\{1,2,3,4\}]}, \\ & s \sqsubseteq \mathbf{[\{3\},\{1,2,3,5\}]} \ \} \end{aligned}$$

Entonces[3]:

$$s \sqsubseteq [\emptyset, \top_{\mathit{Set\ Integer}}] \ \cap_L \ s \sqsubseteq \mathbf{[\{1\},\{1,2,3,4\}]} \ = \ s \sqsubseteq \mathbf{[\{1\},\{1,2,3,4\}]}$$
$$s \sqsubseteq \mathbf{[\{1\},\{1,2,3,4\}]} \ \cap_L \ s \sqsubseteq \mathbf{[\{3\},\{1,2,3,5\}]} \ = \ s \sqsubseteq \mathbf{[\{1,3\},\{1,2,3\}]}$$

[3]Observa que $s ::' \mathit{Set\ Integer}$ es la restricción $s \sqsubseteq [\emptyset, \top_{\mathit{Set\ Integer}}]$.

### 3.6.2. Otros ejemplos

#### Cadenas Binarias

El dominio de las cadenas binarias $\mathcal{B}$ es el conjunto de todas las secuencias (posiblemente infinitas) de ceros y unos junto con el elemento $\top_{\mathcal{B}}$. El elemento bottom $\bot_{\mathcal{B}}$ es la secuencia vacía. Para todo $b_1, b_2 \in \mathcal{B}$, definimos $b_1 \preceq_{\mathcal{B}} b_2$ si y sólo si $b_1$ es un prefijo de $b_2$. Por tanto, $glb_{\mathcal{B}}(b_1, b_2)$ es el prefijo común más largo de $b_1$ y $b_2$ (e.g. $glb_{\mathcal{B}}(00010, 00111) = 00$) y $lub_{\mathcal{B}}(b_1, b_2)$ es $\top_{\mathcal{B}}$ si $b_1 \not\sim b_2$, $b_2$ si $b_1 \preceq_{\mathcal{B}} b_2$ y $b_1$ si $b_2 \preceq_{\mathcal{B}} b_1$. Por tanto

$$(\mathcal{B}, \preceq_{\mathcal{B}}, glb_{\mathcal{B}}, lub_{\mathcal{B}}, \bot_{\mathcal{B}}, \top_{\mathcal{B}})$$

es un retículo. Entonces, si $\mathcal{B}$ se toma como el dominio de computación y definimos $+/2$ como un operador que concatena dos cadenas binarias, las siguientes restricciones de intervalo para $x, y \in V_{\mathcal{B}}$

$$x, y ::' \mathcal{B}, \qquad x \sqsubseteq [001 \overline{+} min(y), \top_{\mathcal{B}}]$$

limitan $x$ a tomar valores en el conjunto de las cadenas binarias que empiezan con la cadena 001.

### 3.6.3. Combinaciones de Dominios

Nuevos dominios de computación pueden ser construidos mediante el uso de operadores clásicos definidos para la combinación de retículos.

#### Productos de dominios

Por ejemplo, considera el retículo *Integer*.
(1) Un punto en un plano puede definirse mediante sus coordenadas Cartesianas usando el producto directo $Point = \langle Integer, Integer \rangle$.
(2) Un rectángulo (donde el eje de abscisas está en paralelo con la base del rectángulo) puede definirse mediante dos puntos (la esquina inferior izquierda y la esquina superior derecha). Sea *Rect* el producto directo $\langle Point, Point \rangle$.

Ahora es posible declarar restricciones de intervalo sobre estos dominios. Por ejemplo, considera $re \in V_{Rect}$, entonces

$$re \sqsubseteq [\langle \langle 2, 2 \rangle, \langle 5, 5 \rangle \rangle, \langle \langle 4, 4 \rangle, \langle 7, 7 \rangle \rangle]$$

restringe la esquina inferior izquierda del rectángulo a tomar valores en el plano $\langle 2, 2 \rangle \times \langle 4, 4 \rangle$ y la esquina superior derecha a tomar valores en el plano $\langle 5, 5 \rangle \times \langle 7, 7 \rangle$.

#### Sumas de Dominios

Considera el retículo *AtoF* que contiene todos los caracteres alfabéticos (en mayúsculas) entre la 'A' y la 'F' con la relación de orden tradicional 'A'<'B'<...<'F', y el retículo *0to9* que contiene los caracteres numéricos desde el '0' hasta el '9' con la relación de orden '0'<'1'<...<'8'<'9'. Entonces el retículo de dígitos hexadecimales *0toF*

puede ser definido como el retículo $\mathit{0to9} \uplus \mathit{AtoF}$ (i.e., como la suma lineal de los retículos $\mathit{AtoF}$ y $\mathit{0to9}$).

## 3.7. Contribuciones

Hemos propuesto una generalización del modelo de indexicals sobre dominios finitos a conjuntos de retículos. La novedad se basa en dos pasos fundamentales:

1. La construcción formal de una estructura de retículos de intervalo sobre la cual las restricciones pueden ser genéricamente propagadas, independientemente de la naturaleza del dominio de computación y de la cardinalidad del mismo.

2. La definición de los operadores de restricción sobre multiples dominios, lo que da lugar a la generación de canales de sentido único a través de los cuales los resolutores definidos sobre dominios diferentes pueden comunicarse, de manera que la información fluye de un dominio de computación a otro.

Además hemos propuesto un esquema genérico para la propagación de las restricciones de intervalo. Otras contribuciones de éste son enumeradas a continuación:

- Nuestro esquema proporciona un mecanismo novedoso para comprobar *a priori* la monotonía de las restricciones de intervalo, que puede usarse sobre cualquier dominio con estructura de retículo y servir para cualquier sistema basado en el modelo indexical.

- Muchos sistemas de restricciones existentes son instancias de nuestro esquema para la propagación. Como consecuencia, nuestro esquema puede explicar el comportamiento operacional de todos estos sistemas y garantizar las propiedades principales tales como la terminación y la corrección.

  Además, nuestro esquema también demuestra que la estructura de retículo de un dominio de computación es el motor principal de muchos sistemas de propagación basados en restricciones.

- Nuestro esquema de propagación es una alternativa al enfoque genérico y transparente, basado en reglas, del lenguaje CHR.

# Capítulo 4

# Cooperación de Resolutores de Intervalo

*Un pour tous et tous pour un.*

Les Trois Mousquetaires (1844)
Alexandre Dumas, 1802-70

## 4.1. Introducción y Motivaciones

En el Capítulo 3, hemos propuesto una alternativa al enfoque transparente de CHR que está basada en el enfoque indexical de clp(FD) (Codognet y Diaz, 1996a). En el capítulo 2 hemos mostrado que este enfoque es bastante eficiente. En nuestra propuesta, hemos definido un esquema para la propagación de las restricciones de intervalo sobre cualquier retículo. Por tanto, este enfoque es válido y aplicable a todos los dominios clásicos como son los Booleanos, los reales, los intervalos finitos de enteros y los conjuntos, así como a otros dominios especializados y construidos para aplicaciones específicas.

Sin embargo en la práctica, las restricciones no suelen ser homogéneas y no se suelen adaptar a un único dominio de computación. Como consecuencia, la formulación de problemas debe adaptarse a los dominios soportados por un sistema de CLP. En realidad, muchos problemas se expresan mejor cuando se pueden utilizar dominios diferentes.

En este capítulo extendemos el esquema para la propagación de restricciones de intervalo anteriormente formalizado, para que admita la cooperación de resolutores (posiblemente definidos sobre dominios diferentes). Para ello, partimos de los fundamentos básicos establecidos en el capítulo 3 donde mostramos que, al definir los elementos del dominio $L^b$, se permitía la definición de un conjunto de operadores que reciben,

como argumentos, elementos pertenecientes a multiples dominios (posiblemente distintos) $L_1^s, \ldots, L_n^s$ y devuelven un elemento de otro dominio $L^s$. Esto, combinado con el concepto de indexical, da lugar a la generación de canales de sentido único (para la trasmisión de la información) que van desde los dominios $L_1^b, \ldots, L_n^b$ al dominio $L^b$.

Una cooperación más general requeriría una comunicación en un doble sentido entre los dominios de computación. Por ello, hemos definido el concepto de *restricción de alto nivel* a ser una relación sobre un dominio que se construye a partir de un conjunto de dominios de computación.

## 4.2. Restricciones de Alto Nivel

Para diferenciar el concepto de restricción de intervalo definido en el capítulo anterior, de las restricciones de alto nivel, definidas en este capítulo, llamaremos en lo sucesivo *restricciones primitivas* a aquellas de la forma $x \sqsubseteq r$.

De la misma forma que en el lenguaje clp(FD) (Codognet y Diaz, 1996a), las restricciones de alto nivel pueden ser definidas en términos de otras restricciones de intervalo (bien sean primitivas o bien sean de alto nivel).

**DEFINICIÓN 4.1** *(Restricción de alto nivel) Supón que* $\mathcal{L}' = \{L_1, \ldots, L_m\} \subseteq \mathcal{L}$. *Entonces* $q :: L_1 \times \ldots \times L_m$ *es una* relación de restricciones $m$-aria *para* $\mathcal{L}'$. *Supón que* $x_1 \in V_{L_1}, \ldots, x_m \in V_{L_m}$, *y* $c_1, \ldots, c_n$ *son restricciones con variables restringidas* $X \supseteq \{x_1, \ldots, x_m\}$. *Entonces*

$$q(x_1, \ldots, x_m) \Leftrightarrow c_1, \ldots, c_n.$$

*es una* restricción de alto nivel *sobre* $\mathcal{L}'$.

A diferencia de clp(FD), nuestro esquema también permite la definición de restricciones genéricas o sobrecargadas. Una restricción de alto nivel es *genérica* para los argumentos $i_1, \ldots, i_j$ $(1 \leq i_1 < \cdots < i_j \leq m)$ si su definición es independiente de la elección de los dominios $L_{i_1}, \ldots, L_{i_j}$ en $\mathcal{L}$. Una restricción está *sobrecargada* para los argumentos $i_1, \ldots, i_j$ si está definida para cualquier $L_{i_1}, \ldots, L_{i_j}$ en $\mathcal{L}_1$ donde $\mathcal{L}_1 \subset \mathcal{L}$ y $\#(\mathcal{L}_1) > 1$.

**EJEMPLO 4.2** *Considera la siguiente restricción de alto nivel:*

$$\begin{aligned} x \leq y \Leftrightarrow x \sqsubseteq [\bot_L, & max(y), \\ y \sqsubseteq [ & min(x), \top_L]. \end{aligned} \tag{4.1}$$

*Entonces esta restricción es genérica para ambos argumentos de* $\leq$, *puesto que cada* $L \in \mathcal{L}$ *tiene elementos top y bottom (posiblemente "liftados").*

**EJEMPLO 4.3** *Considera la definición de los operadores − y + mostrados en el Ejemplo 3.9 y la siguiente definición de una restricción* plus/3*:*

$$\begin{aligned} plus(x,y,z) \Leftrightarrow x &\sqsubseteq min(z) \overline{-} max(y), max(z) - min(y), \\ y &\sqsubseteq min(z) \overline{-} max(x), max(z) - min(x), \\ z &\sqsubseteq min(x) \overline{+} min(y), max(x) + max(y). \end{aligned}$$

*donde $x, y, z \in V_L$ para algún $L \in \mathcal{L}$. Esta restricción esta sobrecargada puesto que es válida para cualquier dominio $L$ en el cual los operadores $+_L$ y $-_L$ están definidos.*

Sea $c \Leftrightarrow c_1, \ldots c_n$ una restricción de alto nivel definida sobre un conjunto de restricciones $C$. Para propagar $C$ con respecto a un almacén de restricciones simple y estable $S$, primero reemplazamos $c$ (en $C$) por $c_1, \ldots, c_n$. Repetimos este paso hasta que $C$ contiene sólo restricciones primitivas. Naturalmente, la terminación no es garantizada y dependerá de las definiciones de las restricciones de alto nivel. Tan pronto como $C$ contenga sólo restricciones primitivas, la propagación de restricciones y la estabilización de las mismas se ejecuta como en el esquema básico mostrado en el capítulo 3.

**EJEMPLO 4.4** *Considera $\mathcal{L} = \{Set\ \Re, \Re\}$, las variables $x, y \in V_{Set\ \Re}$, $z, w \in V_\Re$ y los siguientes almacenes estables y simples:*

$$\begin{aligned} S = \{\ & x \sqsubseteq [\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{0}\}, \{\mathbf{5},\mathbf{6},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{0},\mathbf{7},\mathbf{4},\mathbf{9},\mathbf{3}\}),\ z \sqsubseteq (\mathbf{2},\mathbf{0},\mathbf{15},\mathbf{5}), \\ & y \sqsubseteq [\{\}, \{\mathbf{3},\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{7},\mathbf{4},\mathbf{4},\mathbf{8}\}), \qquad w \sqsubseteq [\mathbf{0},\mathbf{0},\mathbf{12},\mathbf{0}]\ \}; \\ S' = \{\ & x \sqsubseteq [\{\}, \{\mathbf{3},\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{7},\mathbf{4},\mathbf{4},\mathbf{8}\}), \qquad z \sqsubseteq [\bot_\Re, \mathbf{12},\mathbf{0}], \\ & y \sqsubseteq [\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{0}\}, \top_{Set\ \Re}], \qquad w \sqsubseteq (\mathbf{2},\mathbf{0}, \top_\Re]\ \}; \\ S'' = \{\ & x \sqsubseteq [\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{0}\}, \{\mathbf{3},\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{7},\mathbf{4}\}), \qquad z \sqsubseteq (\mathbf{2},\mathbf{0},\mathbf{12},\mathbf{0}], \\ & y \sqsubseteq [\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{0}\}, \{\mathbf{3},\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{7},\mathbf{4},\mathbf{4},\mathbf{8}\}), \qquad w \sqsubseteq (\mathbf{2},\mathbf{0},\mathbf{12},\mathbf{0}]\ \}. \end{aligned}$$

*Considera ahora la restricción de alto nivel $\leq/2$ tal como fue definida en el Ejemplo 4.2. Entonces $\{x \leq y,\ z \leq w\} \rightsquigarrow^S S'$ y $S \cup S' \mapsto S''$.*

**EJEMPLO 4.5** *Supón que $L \in \{Integer, \Re, Point\}$, donde Point se define como en la Sección 3.6.3 (i.e., $Point = \langle Integer, Integer \rangle$), que los operadores + y − son tal como fueron definidos en el Ejemplo 3.9 donde, para los dominios enteros y reales $+_L$ y $-_L$ son definidos como es usual (i.e., devuelven la suma y la diferencia de dos números en L) y en el dominio de los puntos enteros $+_{Point}$ y $-_{Point}$ se define como*

$$\begin{aligned} \langle a,b \rangle +_{Point} \langle c,d \rangle &= \langle a+c, b+d \rangle, \\ \langle a,b \rangle -_{Point} \langle c,d \rangle &= \langle a-c, b-d \rangle, \end{aligned}$$

*que $x, y, z \in V_{Integer}$, que $r, w, t \in V_\Re$, que $p_1, p_2, p_3 \in V_{Point}$ y considera el siguiente almacén de restricciones:*

$$\begin{aligned} S = \{ & t \sqsubseteq [\mathbf{1},\mathbf{0},\mathbf{4},\mathbf{0}],\ w \sqsubseteq (\mathbf{0},\mathbf{0},\mathbf{90},\mathbf{0}], \\ & x \sqsubseteq [\mathbf{1},\mathbf{2}],\ y \sqsubseteq [\mathbf{2},\mathbf{9}), \\ & p_1 \sqsubseteq [\langle 0,0 \rangle, \langle 1,2 \rangle],\ p_2 \sqsubseteq [\langle 1,3 \rangle, \langle 2,9 \rangle]\ \}. \end{aligned}$$

*Considera ahora la restricción plus/3 definida en el Ejemplo 4.3. Esta definición está sobrecargada y es válida para los enteros, los reales y el dominios de los puntos enteros puesto que $+_L$ y $-_L$ están definidos sobre estos dominios. Por ello, esta restricción puede emplearse sobre estos dominios. Por ejemplo,*

$$\{plus(r,w,t)\} \rightsquigarrow^S \{\ r \sqsubseteq [-\mathbf{89},\mathbf{0},\mathbf{4},\mathbf{0})\ \},$$
$$\{plus(x,y,z)\} \rightsquigarrow^S \{\ z \sqsubseteq [\mathbf{3},\mathbf{11})\ \},$$
$$\{plus(p_1,p_2,p_3)\} \rightsquigarrow^S \{\ p_3 \sqsubseteq [\langle 1,3\rangle, \langle 3,11\rangle]\ \}.$$

## 4.3. Ejemplos

En esta sección mostramos un par de ejemplos que demuestran la declaratividad del enfoque cooperativo.

### 4.3.1. Combinaciones Lineales

En los actuales sistemas de CLP, la mayoría de las operaciones aritméticas, ejecutadas sobre un dominio particular, devuelven un valor perteneciente a dicho dominio. Por ejemplo, una expresión tal como $a+b$, donde $a$ y $b$ son reales, devuelve un valor de tipo real. Sin embargo, en la práctica, existen dominios bien conocidos, tales como el dominio de los números binarios o el de los números hexadecimales, en los cuales ésto no es necesariamente cierto (véase la Figura 4.1). Por ejemplo, en el dominio hexadecimal 'F'+'F' no pertenece a este dominio sino que pertenece al producto lexicográfico $(hexadecimal, hexadecimal)$. Esto significa que la definición estándar del operador + no es válida.

Figura 4.1: Combinaciones lineales de la suma de dominios

Una solución tradicional consiste en extender la operación + con un argumento para contener el acarreo. Por ejemplo, F+F=(1,E). A continuación presentamos una solución más elegante que está basada en el concepto de colaboración de resolutores (i.e., combinación de dominios + cooperación).

Sea $0toF$ el retículo de los dígitos hexadecimales tal como se definió en la Sección 3.6.3 (i.e., $0toF = 0to9 \uplus AtoF$) y $0toF^2$ el producto lexicográfico $(0toF, 0toF)$. Ahora sobrecargamos los operadores + y − mostrados en el Ejemplo 3.9 mediante la siguiente definición sobre los hexadecimales

$$+ :: 0toF^s \times 0toF^s \to 0toF^{2s} \qquad - :: 0toF^{2s} \times \overline{0toF^s} \to 0toF^s$$
$$h_1\}_1 + h_2\}_2 = h_1 +_{0toF^2} h_2\ \ \}_1 +_B \}_1 \qquad (h_1,h_2)\}_1 - \{_2 h_3 = (h_1,h_2) -_{0toF} \hat{h_3}\ \ \}_1 +_B \}_2$$

donde $+_B$ se define como en el Ejemplo 3.9 y $+_{0toF^2}$ y $-_{0toF}$ se definen como es usual en el dominio de los hexadecimales i.e.,

$$
\begin{array}{ll}
+_{0toF^2} :: 0toF \times 0toF \to 0toF^2 & -_{0toF} :: 0toF^2 \times \widehat{0toF} \to L \\
\text{`0'} +_{0toF^2} \text{`0'}=(\text{`0'},\text{`0'}), & (\text{`0'},\text{`0'}) -_{0toF} \hat{\text{`0'}} = \text{`0'}, \\
\ldots & \ldots \\
\text{`1'} +_{0toF^2} \text{`F'}=(\text{`1'},\text{`0'}), & (\text{`1'},\text{`0'}) -_{0toF} \hat{\text{`F'}} = \text{`1'}, \\
\ldots & (\text{`1'},\text{`0'}) -_{0toF} \hat{\text{`1'}} = \text{`F'}, \\
\ldots & \ldots, \\
\text{`F'} +_{0toF^2} \text{`F'}=(\text{`1'},\text{`E'}), & (\text{`1'},\text{`E'}) -_{0toF} \hat{\text{`F'}} = \text{`F'}, \\
\ldots & \ldots
\end{array}
$$

Entonces,

$$\text{`1']}+\text{`F']} = (\text{`1'},\text{`0'})] \text{ y } (\text{`1'},\text{`0'})] - [\text{`1'} = \text{`F'}].$$

Como consecuencia, la restricción $plus/3$ definida en el Ejemplo 4.3 puede ser usada sobre el dominio de los hexadecimales, siempre que consideremos $x, y \in V_{0toF}$ y $z \in V_{0toF^2}$.

Por ejemplo, sea $h_1, h_2 \in V_{0toF}$ y $h_3 \in V_{0toF^2}$, y considera el siguiente almacén de restricciones:

$$S = \{\ h_1 \sqsubseteq [\text{`0'},\text{`F'}], h_2 \sqsubseteq (\text{`A'},\text{`F'}]\ \}.$$

Entonces

$$\{\ plus(h_1, h_2, h_3)\ \} \rightsquigarrow^S \{\ h_3 \sqsubseteq [(\text{`0'},\text{`A'}),(\text{`1'},\text{`E'})]\ \}.$$

### 4.3.2. Incluso más Expresividad!

La cooperación puede también ser alcanzada a partir de dominios no-relacionados.

**EJEMPLO 4.6** *Sean $Integer_0$ el conjunto de los enteros no-negativos y $\Re_0$ el conjunto de los reales no-negativos. Considera la siguiente sentencia 'imperativa':*

$$\textbf{si } b \textbf{ entonces } i \leftarrow trunc(r * r) \textbf{ sino } i \leftarrow trunc(r) \textbf{finsi}$$

*donde $b \in Bool$, $r \in \Re_0$ e $i \in Integer_0$. El significado abstracto de esta sentencia condicional puede ser simulado en la teoría de los intervalos mediante la siguiente restricción de intervalo:*

$$c_1 \ = \ i \sqsubseteq min(b)\overline{\rhd}min(r), max(b) \rhd max(r)$$

*donde $r \in V_{\Re_0}$, $b \in V_{Bool}$, $i \in V_{Integer_0}$ y $\rhd$ es un operador de restricción para ${Integer_0}^s$ declarado y definido como sigue:*

$$
\begin{array}{l}
\rhd :: Bool^s \times {\Re_0}^s \to {Integer_0}^s \\
true] \rhd r\} = trunc_{Integer}(r * r)] \\
false] \rhd r\} = trunc_{Integer}(r)].
\end{array}
$$

*donde* $trunc_{Integer}(a)$ *devuelve la parte entera de cualquier elemento* $a \in \Re_0$. *Puesto que el dominio Booleano es discreto, observa que sólo consideramos valores "cerrados" ya que los valores "abiertos" pueden reducirse a éstos mediante la aplicación de las reglas de equivalencia descritas en la Sección 3.4.4.*

*La propagación de restricciones se realiza como en el marco básico. Por ejemplo*

$$S = \{\ r \sqsubseteq [\mathbf{2},\mathbf{3},\mathbf{8},\mathbf{9}), b \sqsubseteq [false, true]\ \}.$$

*Entonces*[1]

$$\{c_1\} \rightsquigarrow^S \{\ i \sqsubseteq [\mathbf{2}, \mathbf{79}]\ \}.$$

*Más propagación puede obtenerse añadiendo más restricciones. Por ejemplo,*

$$S \cup \{b \sqsubseteq [false, false]\} \mapsto S_1$$

*y* $S_1 = \{r \sqsubseteq [\mathbf{2},\mathbf{3},\mathbf{8},\mathbf{9}), b \sqsubseteq [false, false]\ \}$. *Entonces*[2] $\{c_1\} \rightsquigarrow^{S_1} \{\ i \sqsubseteq [\mathbf{2}, \mathbf{8}]\ \}$.

## 4.4. Conclusiones

En este capítulo hemos extendido el esquema básico de propagación descrito en el capítulo anterior con el objeto de producir un mecanismo que permita la propagación entre dominios diferentes. La extensión está basada en la definición de las restricciones de alto nivel como relaciones sobre un dominio de computación construidas a partir de un conjunto de dominios de computación. Como resultado, en el esquema extendido, los resolutores pueden verse como relaciones formuladas sobre uno o más dominios de computación y, por tanto, los resolutores pueden ser construidos sobre dominios muy diferentes o, incluso, puede ser definidos sobre varios dominios.

La flexibilidad del esquema resultante se ha demostrado a través de ejemplos no convencionales.

Además, nuestro esquema cooperativo proporciona tres características que cualquier sistema cooperativo debería tener:

1. *La seguridad de tipo*: nuestro esquema previene la cooperación errónea de resolutores. Observa que los resolutores pueden ser conectados libremente mediante restricciones de intervalo formuladas con operadores definidos sobre varios dominios. Por tanto, sólo las restricciones que son admisibles en el sistema con respecto a las declaraciones de los operadores, son válidas en la formulación de los problemas.

2. *Transparencia en la comunicación de los resolutores*: En nuestro esquema la cooperación viene de definir adecuadamente los operadores de restricción sobre diversos dominios y de permitir que las restricciones de alto nivel definan el mecanismo de propagación entre los diferentes dominios "cooperando". Como ya se

[1] Observa que $i \sqsubseteq [\mathbf{2}, \mathbf{79}]$ es el resultado de evaluar $i \sqsubseteq [false \overline{\rhd} [2,3, true] \rhd 8,9)$.
[2] Observa que $i \sqsubseteq [\mathbf{2}, \mathbf{8}]$ es el resultado de evaluar $i \sqsubseteq [false \overline{\rhd} [2,3, false] \rhd 8,9)$.

ha indicado, tanto los operadores como las restricciones de alto nivel son transparentes desde el punto de vista del usuario, de manera que el esquema cooperativo es transparente.

3. *Colaboración de resolutores*, es decir, combinación de dominios y/o resolutores, y cooperación de éstos últimos.

## 4.5. Contribuciones

En este capítulo hemos presentado un mecanismo nuevo para la cooperación de restricciones en el cual la interacción de los resolutores se alcanza a partir de operadores que se definen sobre dominios diferentes, y a partir de la definición correcta de las restricciones de alto nivel. Como consecuencia, los operadores hacen el rol de interfaz entre los dominios, mientras que las restricciones de alto nivel juegan el papel de 'pipelines' a través de los cuales la información se transmite entre los resolutores (i.e., es 'enviada a' y 'recibida por' los resolutores). A nuestro entender este enfoque es completamente nuevo.

Desde el punto de vista del usuario, los operadores de restricción y las restricciones de alto nivel son totalmente transparentes, así que nuestro sistema cooperativo es un sistema transparente. Con la excepción del lenguaje CHR, es el único lenguaje conocido (por lo que sabemos) que permite a los usuarios definir su propio interfaz entre los resolutores así como el grado de cooperación entre los mismos (es decir, el sentido físico en el que el usuario desea que la información se propague).

# Capítulo 5

# Ramificación de Restricciones de Intervalo

*Nothing is particularly hard if you divide it into small jobs.*

Henry Ford 1863 - 1947

## 5.1. Introducción

En los capítulos previos hemos descrito un esquema de propagación para las restricciones de intervalo que es genérico y que permite resolver (en la mayoría de los casos sólo parcialmente) CSPs (i.e., un conjunto de restricciones de intervalo definidas sobre un conjunto de retículos). Nuestro esquema elimina los valores inconsistentes del dominio inicial de las variables que no pueden llegar a ser parte de ninguna solución. Los resultados se propagan a través de un conjunto de restricciones y el proceso se repite hasta alcanzar un conjunto estable. Sin embargo, aunque nuestro esquema encuentra la *solución más general* al almacén de restricciones representando el (véase el Teorema 3.45), no es completo puesto que podría no determina cuáles son, exactamente, los valores de los intervalos que pertenecen a las soluciones concretas del problema.

Por esta razón, en este capítulo, proponemos un esquema de ramificación que es complementario al esquema de propagación que descrito anteriormente. La combinación de ambos esquemas forma un esquema de resolución completa de las restricciones de intervalo que pueden ser formuladas sobre cualquier conjunto de retículos, de nuevo independientemente de la naturaleza y cardinalidad de éstos. Por ello, este nuevo esquema puede ser usado en la mayoría de los dominios clásicos existentes y, al igual que para el esquema descrito anteriormente, también es válido para sistemas cooperativos y para sistemas soportando diversos dominios.

Además, también describimos un conjunto de propiedades interesantes que son satisfechas por cualquier instancia del esquema, y mostramos que el comportamiento

operacional de muchos sistemas lógicos con restricciones (incluyendo sistemas cooperativos) puede ser explicados completamente por nuestro esquema.

## 5.2. Algunos Conceptos Importantes

Continuamos usando $L$ para denotar cualquier dominio en $\mathcal{L}$, $X \in \wp_f(\mathcal{V}_\mathcal{L})$ el conjunto de variables restringidas, $\mathcal{C}_L^X$ el conjunto de todas las restricciones de intervalo para $L$ con variables restringidas en $X$, $\mathcal{C}^X$ el dominio de todas las restricciones de intervalo sobre $X$ y $\mathcal{SS}^X$ el conjunto de todos los almacenes de restricciones simples y estables para $X$. También $L_<$ denota cualquier retículo totalmente ordenado en $\mathcal{L}$.

**Notación.** Si $\{c_1, \ldots, c_n\} \in \mathcal{SS}^X$ y $i \in \{1 \ldots, n\}$, entonces

$$\{c_1, \ldots, c_n\}[c_i/c'] = \{c_1, \ldots, c_{i-1}, c', c_{i+1}, \ldots, c_n\}.$$

**DEFINICIÓN 5.1** *(Divisibilidad) Sea $c = x \sqsubseteq \overline{s}, t$ una restricción de intervalo consistente en $\mathcal{C}_L^X$. Entonces, $c$ es* divisible *si $s \neq_{L^s} t$ y* no-divisible *en cualquier otro caso. Sea $S \in \mathcal{SS}^X$ un almacén de restricciones consistente. Entonces $S$ es* divisible *si existe $c \in S$ tal que $c$ es divisible y* no-divisible *en otro caso.*

Observa que, por la Definición 3.17, una restricción no-divisible tiene la forma $x \sqsubseteq$ **[a,a]** la cual, como se dice en la Definición 3.21 equivale a $x =$ a donde $x \in V_L$ y a $\in L$ para algún $L \in \mathcal{L}$ (i.e., una restricción no-divisible puede ser vista como un asignación de una variable a un valor).

**EJEMPLO 5.2** *Sean $x, y \in V_{Integer}$, $r, w \in V_\Re$ y $S, S' \in \mathcal{SS}^{\{x,r\}}$. Entonces,*

$$\begin{array}{llll} x \sqsubseteq [\mathbf{1}, \mathbf{4}] & y & r \sqsubseteq (\mathbf{1},\mathbf{0}, \mathbf{3},\mathbf{2}] & \textit{son divisibles;} \\ y \sqsubseteq [\mathbf{2}, \mathbf{2}] & y & w \sqsubseteq [\mathbf{1},\mathbf{5}, \mathbf{1},\mathbf{5}] & \textit{son no-divisibles.} \end{array}$$

*También*

$$\begin{array}{ll} S = \{\ x \sqsubseteq [\mathbf{1}, \mathbf{1}],\ r \sqsubseteq [\mathbf{1},\mathbf{0}, \mathbf{1},\mathbf{0}]\ \} & \textit{es no-divisible;} \\ S' = \{\ x \sqsubseteq [\mathbf{1}, \mathbf{4}],\ r \sqsubseteq [\mathbf{1},\mathbf{0}, \mathbf{1},\mathbf{0}]\ \} & \textit{es divisible.} \end{array}$$

Observa que otros caso de 'no-divisibilidad' pueden ser detectados mediante la equivalencia de rangos sobre los dominios discretos discutida en la Sección 3.4.4.

**EJEMPLO 5.3** *Considera $L = Integer$ y la restricción consistente $c = x \sqsubseteq (\mathbf{1}, \mathbf{2}]$. Entonces $c$ es no-divisible puesto que el rango $(\mathbf{1}, \mathbf{2}]$ es equivalente al rango $[\mathbf{2}, \mathbf{2}]$ en $R_L^s$ y la restricción $c = x \sqsubseteq [\mathbf{2}, \mathbf{2}]$ es no-divisible.*

**PROPOSICIÓN 5.4** *Sea $X \in \wp_f(V_\mathcal{L})$.*

*(1) Sean también $c, c' \in \mathcal{C}_L^X$ tal que $c \prec_{\mathcal{C}_L^X} c'$. Entonces, si $c$ es consistente, $c'$ es divisible.*

*(2) Sean también $S, S' \in \mathcal{SS}^X$ tal que $S \prec_s S'$. Entonces, si $S$ es consistente, $S'$ es divisible.*

En la Sección 3.4.5 nosotros definimos el concepto de *solución* para un almacén de restricciones como un almacén consistente y estable que no puede ser reducido más por el proceso de propagación. Ahora redefinimos este concepto para capturar el concepto tradicional de solución como una asignación de las variables restringidas a valores de los dominios de computación de manera que todas las restricciones sean satisfechas. De esta forma, para diferenciar lo que es una solución, tal como se definió en la Sección 3.4.5, del concepto definido en este capítulo, a partir de ahora usamos el término *solución* para referirnos al concepto anteriormente definido y el término *solución auténtica* para referirnos al concepto definido en este capítulo.

**DEFINICIÓN 5.5** *(Solución auténtica) Sea $C \in \wp_f(\mathcal{C}^X)$ un almacén de restricciones consistente para $X$ y $R \in \mathcal{SS}^X$. Entonces, $R$ es una* solución auténtica para $C$ *si $R$ es no-divisible y además una solución para $C$.*

*$R' \in \mathcal{SS}^X$ es una* solución parcial para $C$ *si existe una solución auténtica $R''$ para $C$ tal que $R'' \prec_s R'$. En este caso decimos que $R'$* cubre *$R''$.*

**EJEMPLO 5.6** *Considera los operadores $+$ y $-$ para $L = Integer$ tal como fueron definidos en el Ejemplo 3.9, $x, y \in V_{Integer}$, $X = \{x, y\}$, $C \in \wp_f(\mathcal{C}^X)$ donde*

$$\begin{aligned} C \;=\; \{\; & x \sqsubseteq [0, max(y) - [1, \\ & y \sqsubseteq [1 + min(x), 100] \;\} \end{aligned}$$

*y $S, S' \in \mathcal{SS}^X$ donde*

$$\begin{aligned} S \;&=\; \{\; x \sqsubseteq [\mathbf{1}, \mathbf{4}],\; y \sqsubseteq [\mathbf{2}, \mathbf{5}] \;\}, \\ S' \;&=\; \{\; x \sqsubseteq [\mathbf{1}, \mathbf{1}],\; y \sqsubseteq [\mathbf{3}, \mathbf{3}] \;\}. \end{aligned}$$

*Entonces, $S$ es una solución (y también una solución parcial) para $C$ mientras que $S'$ es una solución auténtica para $C$.*

El conjunto de soluciones auténticas para $C$ se denota como $Sol_a(C)$.

**DEFINICIÓN 5.7** *(Pila de almacenes de restricciones) Sea $P = (S_1, \ldots, S_\ell)$ cualquier secuencia (posiblemente vacía) donde $S_i \in \mathcal{SS}^X$ para $1 \leq i \leq \ell$ y $\ell \geq 0$. Entonces $P$ es una* pila de almacenes de restricciones para $X$ *si la operación coloca/2 sobre $P$ está definida para cualquier $S \in \mathcal{SS}^X$ tal como sigue*

*Precondición:* $\{\; P = (S_1, \ldots, S_\ell) \;\}$
$\quad coloca(P, S)$
*Poscondición:* $\{\; P = (S_1, \ldots, S_\ell, S_{\ell+1}),\; S_{\ell+1} = S \text{ y } P \in Pilas(X) \;\}$

*donde Pilas(X) es el conjunto de todas las pilas de almacenes de restricciones para X, y la operación cima/1 sobre P se define como:*

$$\begin{aligned} &\textit{Precondición: } \{P = (S_1, \ldots, S_\ell) \textit{ y } \ell > 0\} \\ &\quad \textit{cima}(P) = S \\ &\textit{Poscondición: } \{S = S_\ell\}. \end{aligned}$$

*Sea otra pila $P' = (S'_1, \ldots, S'_{\ell'}\} \in Pilas(X)$. Entonces $P \preceq_p P'$ si y sólo si para todo $S_i \in P$ $(1 \leq i \leq \ell)$, existe $S'_j \in P'$ $(1 \leq j \leq \ell')$ tal que $S_i \preceq_s S'_j$. En este caso decimos que $P'$* cubre *$P$.*

## 5.3. El Proceso de Ramificación

La ramificación de restricciones normalmente involucra dos pasos de elección que suelen llamarse *ordenación de variables* y *ordenación de valores*. El primer paso selecciona una variable restringida y el segundo divide el dominio asociado a esta variable en una o más partes, introduciendo un punto de elección en el árbol de búsqueda. A continuación explicamos estos dos pasos de elección a partir de la descripción de las funciones que los definen.

La *función de selección* es una heurística para la ordenación de variables.

**DEFINICIÓN 5.8** *(Función de selección) Sea $S = \{c_1, \ldots, c_n\} \in \mathcal{SS}^X$. Entonces*

$$\textit{selecciona} :: \{S \in \mathcal{SS}^X \mid S \textit{ es divisible}\} \to \mathcal{C}^X$$

*es una* función de selección para *$X$ si $\textit{selecciona}(S) = c_j$ donde $1 \leq j \leq n$ y $c_j$ es divisible.*

**EJEMPLO 5.9** *Supón que $X = \{x_1, \ldots, x_n\}$ es un conjunto de variables restringido respectivamente en $L_1, \ldots, L_n \in \mathcal{L}$ y que $S = \{c_1, \ldots, c_n\} \in \mathcal{SS}^X$ es cualquier almacén divisible de restricciones de intervalo para $X$ donde para todo $i \in \{1, \ldots, n\}$, $c_i$ es la restricción de intervalo simple en $S$ con variable restringida $x_i$. A continuación mostramos una función de selección básica que selecciona la restricción de intervalo divisible "la más a la izquierda" en $S$.*

$$\begin{aligned} &\textit{Precondición: } \{S = \{c_1, \ldots, c_n\} \in \mathcal{SS}^X \textit{ es divisible}\} \\ &\quad \textit{selecciona}_{naive}(S) = c_j \\ &\textit{Poscondición: } \{j \in \{1, \ldots, n\},\ c_j \textit{ es divisible y} \\ &\qquad\qquad \forall i \in \{1, \ldots, j-1\} : c_i \textit{ es no-divisible}\}. \end{aligned}$$

En el proceso de ramificación, algunas restricciones necesitan ser particionadas, en dos o más partes, para introducir un punto de elección. Nosotros definimos una *función de partición* la cual proporciona una heurística para la ordenación de valores.

**DEFINICIÓN 5.10** *(Función de partición) Sea $L \in \mathcal{L}$ y $k > 1$. Entonces*

$$divide_L :: \mathcal{C}_L^X \rightarrow \underbrace{\mathcal{C}_L^X \times \ldots \times \mathcal{C}_L^X}_{k \ times}$$

*es una* función de partición $k$-aria para $L$ *si, para todo $c \in \mathcal{C}_L^X$, con $c$ divisible, esta función está definida como $divide_L(c) = (c_1, \ldots, c_k)$ tal que las siguientes propiedades se mantienen*

$$\begin{array}{l} \textit{Completitud}: \ \forall c' \prec_{\mathcal{C}_L^X} c \text{ con } c' \text{ no-divisible}, \exists i \in \{1, \ldots, k\} \,.\, c' \preceq_{\mathcal{C}_L^X} c_i. \\ \textit{Convergencia}: \ c_i \prec_{\mathcal{C}_L^X} c, \ \forall i \in \{1, \ldots, k\}. \end{array}$$

**EJEMPLO 5.11** *Sea $X = \{i, b, r, s\}$, $i \in V_{Integer}$, $b \in V_{Bool}$, $r \in V_{\Re}$ y $s \in V_{Set\ Integer}$ y sean*[1] *$i \sqsubseteq [\mathbf{a}, \mathbf{a}']$, $b \sqsubseteq [false, true]$, $r \sqsubseteq \{\mathbf{c}, \mathbf{d}\}$ y $s \sqsubseteq \{\mathbf{e}, \mathbf{f}\}$ restricciones divisibles en $\mathcal{C}^X$ donde $\mathbf{a}, \mathbf{a}' \in Integer$, $\mathbf{c}, \mathbf{d} \in \Re$ y $\mathbf{e}, \mathbf{f} \in Set\ Integer$. Entonces, las siguientes funciones son funciones binarias de partición respectivamente para los dominios Integer, Bool, $\Re$ y Set Integer*

$$\begin{array}{l} divide_{Integer}(\ x \sqsubseteq [\mathbf{a}, \mathbf{a}']) = (\ x \sqsubseteq [\mathbf{a}, \mathbf{a}],\ x \sqsubseteq [\mathbf{a} + \mathbf{1}, \mathbf{a}']\ ), \\ divide_{Bool}(\ b \sqsubseteq [false, true]\ ) = (\ b \sqsubseteq [false, false], b \sqsubseteq [true, true]\ ), \\ divide_{\Re}(r \sqsubseteq \{\mathbf{c}, \mathbf{d}\}) \ = \ (\ r \sqsubseteq \{\mathbf{c}, \mathbf{c}'),\ r \sqsubseteq [\mathbf{c}', \mathbf{d}\}\ ), \\ divide_{Set\ Integer}(s \sqsubseteq \{\mathbf{e}, \mathbf{f}\}) = (\ s \sqsubseteq \{\mathbf{e}, \mathbf{f} \backslash \mathbf{g}],\ s \sqsubseteq [\mathbf{e} \cup \mathbf{g}, \mathbf{f}\}\ ). \end{array}$$

*Aquí, $divide_{Integer}$ es una estrategia básica de enumeración de valores desde el mínimo hasta el máximo; $divide_{Bool}$ particiona la única restricción divisible (en el dominio Booleano) en dos restricciones no-divisibles; $divide_{\Re}$ parte el intervalo por el punto medio $\mathbf{c}' = \frac{\mathbf{c}+\mathbf{d}}{\mathbf{2,0}}$; y $divide_{Set\ Integer}$ es una función de partición válida para el dominio de los conjuntos de enteros siempre que definamos $\mathbf{g} = \{\mathbf{l}\}$ y $\mathbf{l} \in \mathbf{f} \backslash \mathbf{e}$.*

**LEMA 5.12** *Sean selecciona/1 una función de selección para $X$, $C \in \wp_f(\mathcal{C}^X)$, $S = (c_1, \ldots, c_n) \in \mathcal{SS}^X$ un almacén divisible de restricciones, $c_j = selecciona(S)$, $c_j \in \mathcal{C}_L^X$ para algún $L \in \mathcal{L}$, $divide_L/1$ una función de partición $k$-aria para $L$ y $(c_{j1}, \ldots, c_{jk}) = divide_L(c_j)$. Entonces*

*(a) $\forall i \in \{1, \ldots, k\} : S[c_j/c_{ji}] \prec_s S$;*

*(b) si $S' \in Sol_a(C)$ y $S' \prec_s S$, entonces*

$$\exists i \in \{1, \ldots, k\} : S' \preceq_s S[c_j/c_{ji}].$$

[1] Observa que, de nuevo, sólo consideramos los intervalos cerrados en el caso de los dominios discretos (i.e., los enteros y los Booleanos). Esto es debido a que estos dominios son reducidos a intervalos cerrados por la equivalencia mostrada en la Sección 3.4.4. Por ejemplo, $x \sqsubseteq (\mathbf{1}, \mathbf{8})$ equivale a $x \sqsubseteq [\mathbf{2}, \mathbf{7}]$.

### 5.3.1. La Función de Precisión como una Regla de Normalización

La función de precisión descrita en la Sección 3.5.2 proporciona un instrumento para la normalización de las funciones de partición, lo que es muy útil en los sistemas que soportan diversos dominios (i.e., sistemas cooperativos)

**EJEMPLO 5.13** *Básicamente, el* principio del primer-fallo *(Haralick y Elliot, 1980) consiste en seleccionar la variable restringida que tiene asociado el menor dominio. Sin embargo, en los sistemas de múltiples dominios, no está siempre claro cuál es el menor dominio (sobre todo si hay varios dominios infinitos). Una forma de medir el tamaño de dominios distintos consiste en normalizarlos mediante la función precision/1 y luego comparar los valores normalizados.*

*Por ejemplo, supón que $X = \{x_1, \ldots, x_n\}$ es un conjunto de variables restringidas, respectivamente, en $L_1, \ldots, L_n \in \mathcal{L}$ y que $S = \{c_1, \ldots, c_n\} \in \mathcal{SS}^X$ es cualquier almacén de restricciones divisible para $X$ donde para cada $i \in \{1, \ldots, n\}$, $c_i$ es la restricción de intervalo simple en $S$ con variable restringida $x_i$. Entonces el principio del primer-fallo puede ser emulado mediante la función de partición selecciona/1 mostrada abajo, la cual selecciona la restricción con la menor precisión*[2]*. Denotamos esta función por $selecciona_{ff}$.*

*Precondición:* $\{S = \{c_1, \ldots, c_n\} \in \mathcal{SS}^X$ *es divisible*$\}$
$selecciona_{ff}(S) = c_j$
*Poscondición:* $\{j \in \{1, \ldots, n\}, c_j$ *es divisible y*
$\forall i \in \{1, \ldots, n\} \backslash \{j\} : c_i$ *divisible* $\implies$ $precision_{L_j}(c_j) \leq_{\Re\mathcal{I}} precision_{L_i}(c_i)\}$.

## 5.4. Ramificación en la Resolución de las Restricciones de Intervalo

La Figura 5.1 muestra un esquema genérico para resolver completamente las restricciones de intervalo. Este esquema complementa el de la Sección 3.5.

El esquema requiere los siguientes parámetros: un conjunto finito $C \in \wp_f(\mathcal{C}^X)$ de restricciones de intervalo a ser resueltas, una almacén de restricciones $S \in \mathcal{SS}^X$ y una cota $p \in \Re\mathcal{I}$. También se necesita una cota real no-negativa $\alpha$. El esquema hace uso del esquema operacional $propaga_\varepsilon/2$ descrito en la Sección 3.5.1. A continuación declaramos más propiedades sobre el esquema de propagación $propaga_\varepsilon/2$.

**LEMA 5.14** *(Más propiedades para $propaga_\varepsilon/2$) Sean $C \in \wp_f(\mathcal{C}^X)$, $S, S^f \in \mathcal{SS}^X$ y $\varepsilon \in \Re^+ \cup \{0{,}0\}$. Supón que $S^f$ es el valor de $S$ después de una ejecución que termina del procedimiento $propaga_\varepsilon(C, S)$. Entonces,*

*(a)* $S^f \preceq_s S$;

[2]Es muy sencillo incluir otras condiciones e.g., si $c_i, c_k, c_j$ tienen la misma precisión (mínima), podemos escoger "más a la izquierda" i.e., $c_{minimum(i,k,j)}$.

*(b)* $\forall R \in Sol_a(C \cup S) : R \preceq_s S^f$.

*(c) Si* $Sol_a(C \cup S)$ *no es el conjunto vacío y* $S^f$ *es no-divisible entonces* $S^f \in Sol_a(C \cup S)$.

*(d) Si* $\varepsilon = 0{,}0$ *y* $S^f$ *es no-divisible entonces* $S^f \in Sol_a(C \cup S)$.

La propiedad (a) asegura que el procedimiento de propagación nunca gana valores nuevos, (b) garantiza que no se pierde ninguna solución, cubierta por el almacén $S$, durante la propagación y las propiedades (c) y (d) garantizan la corrección de las soluciones auténticas.

Además hay un número de valores y procedimientos externos que deben ser definidos externamente al procedimiento de ramificación mostrado en la Figura 5.1:

- una función de selección *selecciona*/1 para $X$;
- una función k-aria de partición $divide_L$ para cada dominio $L \in \mathcal{L}$ (para algún entero $k > 1$);
- una función de precisión para cada $L \in \mathcal{L}$;
- una pila de almacenes de restricciones $P$ para $X$.

Asumimos que los procedimientos externos tienen una implementación que garantiza la terminación para todos los valores posibles.

**TEOREMA 5.15** *(Propiedades del esquema* $resuelve_\alpha/3$*) Sean* $C \in \wp_f(\mathcal{C}^X)$, $S \in \mathcal{SS}^X$, $\varepsilon, \alpha \in \Re^+ \cup \{0{,}0\}$ *y* $p = \top_{\Re\mathcal{I}}$. *Entonces, las siguientes propiedades son garantizadas:*

1. Terminación*: si* $\alpha > 0{,}0$ *y el procedimiento* $propaga_\varepsilon/2$ *termina para todos los valores*[3] *entonces* $resuelve_\alpha(C, S, p)$ *termina;*
2. Completitud: *si* $\alpha = 0{,}0$ *y la ejecución de* $resuelve_\alpha(C, S, p)$ *termina, entonces (en su estado final) la pila* $P$ *contiene todas las soluciones auténticas para* $C \cup S$*;*
3. Completitud aproximada: *si la ejecución de* $resuelve_\alpha(C, S, p)$ *termina y* $R \in Sol_a(C \cup S)$*, entonces (en su estado final) la pila* $P$ *contiene* $R$ *o bien una solución parcial* $R'$ *que cubre* $R$.
4. Corrección: *si* $\alpha = 0{,}0$ *y* $\varepsilon = 0{,}0$*, la pila* $P$ *está inicialmente vacía y la ejecución de* $resuelve_\alpha(C, S, p)$ *termina conteniendo* $R$ *en el estado final de la pila* $P$*, entonces* $R \in Sol_a(C \cup S)$.

[3] Observa que esto se garantiza si $\varepsilon > 0{,}0$ -véase el Teorema 3.53.

procedimiento $resuelve_\alpha(C, S, p)$
empieza
  $propaga_\varepsilon(C, S)$; (1)
  si $S$ es consistente entonces (2)
    si ($S$ es no-divisible o $p < \top_{\Re\mathcal{I}}$ y $p - precision(S) \leq (\alpha, 0)$) entonces (3)
      $coloca(P, S)$; (4)
    sino (5)
      $c_j \leftarrow selecciona(S)$; (6)
      $(c_{j1}, \ldots, c_{jk}) \leftarrow divide_{L_j}(c_j)$, donde $c_j \in \mathcal{C}^X_{L_j}$ y $L_j \in \mathcal{L}$; (7)
      $resuelve_\alpha(C, S[c_j/c_{j1}], precision(S))$ $\vee$
      $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\vee$
      $resuelve_\alpha(C, S[c_j/c_{jk}], precision(S))$; } %% Puntos de Elección (8)
    finsi;
  finsi;
finempieza.

Figura 5.1: $resuelve_\alpha/3$: un esquema genérico de ramificación para la resolución de las restricciones de intervalo

5. Corrección aproximada o control en la precisión de los resultados: *Si $P_{\alpha_1}$ y $P_{\alpha_2}$ son dos pilas no-vacías pertenecientes a $Pilas(X)$ que son el resultado de cualquier ejecución que termina de $resuelve_\alpha(C, S, p)$ (donde inicialmente $P$ está vacía) cuando $\alpha$ toma los valores $\alpha_1$ y $\alpha_2$, respectivamente, y $\alpha_1 < \alpha_2$ entonces*

$$P_{\alpha_1} \preceq_p P_{\alpha_2}.$$

*(En otras palabras, el conjunto de soluciones (posiblemente parciales), en el estado final de la pila, depende del valor de $\alpha$ en el sentido de que cuanto menor sea $\alpha$ mejor será el conjunto de soluciones.)*

Observa que, al igual que la cota $\varepsilon$ en el esquema $propaga_\varepsilon/2$, la cota $\alpha$ también garantiza la terminación y permite controlar la precisión de los resultados.

## 5.5. Problemas de Optimización (COPs)

El esquema de la Figura 5.1 puede adaptarse para resolver problemas de optimización mediante la introducción de tres funciones adicionales.

**DEFINICIÓN 5.16** *(Funciones y valores subsidiarios) Sea $L_< \in \mathcal{L}$ cualquier domi-*

*nio totalmente ordenado*[4] *en* $\mathcal{L}$*. Entonces definimos*

- *una* función de coste, $fcost :: \mathcal{SS}^X \to L_<$;
- *una* relación de orden, $\diamond :: L_< \times L_< \in \{>, <, =\}$;
- *una* cota de coste, $\delta \in L_<$.

Entonces el *esquema de ramificación extendido*, $resuelve_{\alpha+}/3$, se obtiene a partir del esquema $resuelve_\alpha/3$ donde la Línea 4 en la Figura 5.1 es reemplazada por la siguiente sentencia:

$$\begin{array}{ll} \text{si } fcost(S) \diamond \delta \text{ entonces} & (4^*) \\ \quad \delta \leftarrow fcost(S); & \\ \quad coloca(P, S); & \\ \text{finsi;} & \end{array}$$

**TEOREMA 5.17** *(Propiedades del esquema* $resuelve_{\alpha+}/3$*) Sean* $C \in \wp_f(\mathcal{C}^X)$*,* $S \in \mathcal{SS}^X$*,* $\varepsilon, \alpha \in \Re^+ \cup \{0{,}0\}$ *y* $p = \top_{\Re\mathcal{I}}$*. Supón que el procedimiento* $propaga_\varepsilon/2$ *termina para todos los valores*[5]*. Entonces, las siguientes propiedades son garantizadas:*

1. Terminación*: si* $\alpha > 0{,}0$ *entonces la ejecución de* $resuelve_{\alpha+}(C, S, p)$ *termina;*
2. *Si fcost es una función constante con valor* $\delta$ *y* $\diamond$ *es* $=$*, entonces todas la propiedades mostradas en el Teorema 5.15 se mantienen en la ejecución de* $resuelve_{\alpha+}(C, S, p)$*.*
3. Solidez en la optimización*: Si al menos existe una solución para* $C \cup S$ *con coste mayor que* $\bot_{L_<}$ *(resp. menor que* $\top_{L_<}$*),* $\alpha = 0{,}0$*,* $\diamond$ *es* $>$ *(resp.* $<$*),* $\delta = \bot_{L_<}$ *(resp.* $\top_{L_<}$*), la pila* $P$ *está inicialmente vacía y la ejecución de* $resuelve_{\alpha+}(C, S, p)$ *termina con* $P$ *no-vacía, entonces la cima de* $P$ *contiene la primera solución auténtica que maximiza (resp. minimiza) la función de coste.*

Desafortunadamente, si $\alpha > 0{,}0$, no podemos garantizar que la cima de la pila contenga una solución auténtica o incluso una solución parcial para el problema de optimización. Sin embargo, si la función de coste es monótona podemos comparar las soluciones.

**TEOREMA 5.18** *(Solidez aproximada) Supón que, para* $i \in \{1, 2\}$*,* $P_{\alpha_i}$ *es la pila resultante de la ejecución de* $resuelve_{\alpha_i+}(C, S, p)$ *donde* $\alpha_i \in \Re^+ \cup \{0{,}0\}$*. Entonces, si* $\alpha_1 < \alpha_2$ *la siguiente propiedad se mantiene.*

*Si* $P_{\alpha_1}$ *y* $P_{\alpha_2}$ *son no-vacías, y* $cima(P_{\alpha_2})$ *es una solución auténtica o cubre una solución auténtica para* $C \cup S$*, entonces, si* $fcost/1$ *es monótona y* $\diamond$ *es* $<$ *(i.e., es un problema de minimización),*

$$fcost(cima(P_{\alpha_1})) \preceq_{L_<} fcost(cima(P_{\alpha_2})),$$

[4]Normalmente $L_<$ será $\Re$.

[5]De nuevo observa que esto se garantiza siempre que $\varepsilon > 0{,}0$ -véase el Teorema 3.53.

*y, si fcost*/1 *es anti-monótona y* $\diamond$ *es* $>$ *(i.e., es un problema de maximización),*

$$fcost(cima(P_{\alpha_1})) \succeq_{L_<} fcost(cima(P_{\alpha_2})).$$

Una consecuencia directa de este teorema es que si usamos una función de coste (anti-)monótona, cuanto menor sea $\alpha$, mejor será la solución (la cual es una solución probable[6].). Más aún, decrementar $\alpha$ es una forma de descartar soluciones aproximadas. Por ejemplo, en un problema de minimización, si

$$fcost(cima(P_{\alpha_1})) \succ_{L_<} fcost(cima(P_{\alpha_2}))$$

donde *fcost*/1 es monótona, entonces, por la propiedad de la *solidez aproximada* se deduce que $cima(P_{\alpha_2})$ no puede ser una solución auténtica ni puede cubrir una solución auténtica.

### 5.5.1. Formas Diversas de Resolver las Instancias

En esta sección, explicamos cómo instanciando los parámetros *fcost*, $\delta$ y $\diamond$ de formas diferentes el esquema $resuelve_{\alpha+}/3$ se resuelve también de maneras diferentes.

El Teorema 5.17(2) muestra que para resolver CSPs de forma clásica, *fcost* tiene que ser una función constante[7] $\delta$ y $\diamond$ debe tener el valor $=$. Más aún, el Teorema 5.17(3) nos indica que un CSP puede ser resuelto como problema de optimización si instanciamos $\diamond$ como $>$ para problemas de minimización (para problemas de maximización como $<$). En todos los casos, el valor de $\delta$ tiene que ser instanciado al valor inicial de coste a partir del cual se requiere que una solución óptima sea encontrada. Algunas posibles instanciaciones se muestran en la Table 5.1 donde la Columna 1 caracteriza el tipo de CSP, la Columna 2 indica cualquier condición que la función de coste debe cumplir y la Columna 3 su rango (normalmente éste es $\Re$), la Columna 4 da la definición inicial del operador $\diamond$ y la Columna 5 el valor inicial para $\delta$.

| Tipo de CSP | *fcost* | $L_<$ | $\diamond$ | $\delta$ |
|---|---|---|---|---|
| CSP Clásico | constante | $\Re$ | $=$ | $fcost(S)$ |
| Típico COP (Minimización) | cualquier función de coste | $\Re$ | $<$ | $\top_{\Re}$ |
| Típico COP (Maximización) | cualquier función de coste | $\Re$ | $>$ | $\bot_{\Re}$ |
| Max-Min COP | cualquier función de coste | $\Re \times \Re$ | $<$ | $\top_{\Re\times\Re}$ |

Cuadro 5.1: El tipo de CSP depende de la instanciación de los parámetros

Al contrario que los típicos problemas de optimización que normalmente mantienen un criterio fijo (i.e., de maximización o de minimización) y una única cota inferior o superior simple, nuestro esquema también permite combinar criterios (e incluso dar prioridad a unos criterios sobre otros). Este es el caso (véase la Fila 4 en la Tabla 5.1) cuando $L_<$ es un dominio compuesto y la relación de orden en $L_<$ determina la forma en la que el problema se resuelve.

[6]No podemos garantizar que sea una solución auténtica. Ni siquiera que cubra una solución auténtica
[7]Normalmente $\delta \in \Re$.

**EJEMPLO 5.19** *Sean $C \in \wp_f(\mathcal{C}^X)$ un conjunto de restricciones de intervalo a resolver como un COP, $L_<$ el dominio $\Re^2 = \Re \times \Re$ con la relación de orden*

$$(a, b) < (c, d) \iff (a < c \wedge b \geq d) \vee (a \leq c \wedge b > d),$$

*y $fcost :: \mathcal{SS}^X \to L_<$ es una función de coste sobre $\Re^2$ definida para cualquier $S \in \mathcal{SS}^X$ de la siguiente manera*

$$fcost(S) = (fcost_1(S), fcost_2(S))$$

*donde $fcost_1, fcost_2 :: \mathcal{SS}^X \to \Re$ son funciones de coste definidas sobre $\Re$. Entonces, si $\delta$ y $\diamond$ se instancian, respectivamente, $a <$ y $\top_{\Re^2}$ (tal como se muestra en la Fila 4 de la Tabla 5.1), $C$ se resuelve minimizando $fcost_1$ y maximizando $fcost_2$.*

*Por otro lado, si $<$ se define lexicográficamente, es decir,*

$$(a, b) < (c, d) \iff a < c \vee a = c \wedge b < d,$$

*entonces $C$ es resuelto priorizando el criterio de minimización de $fcost_1$ sobre el criterio de minimización de $fcost_2$.*

*Por ejemplo, supón que $Sol_a(C) = \{S_1, S_2, S_3\}$ y $fcost(S_1) = (1{,}0, 5{,}0)$, $fcost(S_2) = (3{,}0, 1{,}0)$ y $fcost(S_3) = (1{,}0, 8{,}0)$. Supón también que estas soluciones han sido encontradas mediante la ejecución del esquema $resuelve_{\alpha+}/3$ con $\diamond \equiv <$ e, inicialmente, $\delta \equiv \top_{\Re^2}$ y que la secuencia en la que las soluciones han sido encontradas es $(S_1, S_2, S_3)$.*

*Considera ahora el primer orden definido arriba sobre $\Re^2$. Cuando $S_1$ es encontrado, la Línea 4* del esquema se ejecuta con $\delta = (\top_\Re, \bot_\Re)$ (i.e., con $\delta = \top_{\Re^2}$ tal como se muestra en la Fila 4 de la Tabla 5.1) y, como consecuencia, $S_1$ es colocado en la pila $P$. A continuación, $S_2$ es encontrada y la Línea 4* se ejecuta con $\delta = fcost(S_1) = (1{,}0, 5{,}0)$. Puesto que $fcost(S_2) \not< (1{,}0, 5{,}0)$, $S_2$ no es colocado en la pila. Después $S_3$ es encontrada y, de nuevo la Línea 4* es ejecutada con $\delta = fcost(S_1) = (1{,}0, 5{,}0)$. Puesto que $fcost(S_3) < (1{,}0, 5{,}0)$ entonces $S_3$ es colocada en la cima de la pila. Observa que $S_3$ minimiza la primera componente y maximiza la segunda.*

*Considera ahora el orden lexicográfico definido sobre $\Re^2$. Cuando $S_1$ es encontrada, la Línea 4* se ejecuta con $\delta = (\top_\Re, \top_\Re)$ (i.e., con $\delta = \top_{\Re^2}$ tal como se muestra en la Fila 4 de la Tabla 5.1) y, como consecuencia, $S_1$ es colocada en la pila $P$. A continuación, $S_2$ es encontrada y La Línea 4* se ejecuta con $\delta = fcost(S_1) = (1{,}0, 5{,}0)$. Como $fcost(S_2) \not< (1{,}0, 5{,}0)$ entonces $S_2$ no se coloca en la pila. Finalmente $S_3$ es encontrada y, de nuevo, la Línea 4* es ejecutada con $\delta = fcost(S_1) = (1{,}0, 5{,}0)$. Puesto que $fcost(S_3) \not< (1{,}0, 5{,}0)$, $S_3$ no se coloca en la pila y la cima de la pila contiene $S_1$. En este caso, $S_1$ es la solución que minimiza prioritariamente el primer componente y, a igualdad de segundos componentes, intenta minimizar la segunda componente.*

## 5.6. Conclusiones

En este capítulo hemos generalizado el método de 'split-and-solve' del sistema CLP(BNR) (Benhamou y Older, 1997) que era aplicable al dominio Booleano, real

y al entero, a cualquier dominio con estructura de retículo, lo que significa que nuestra generalización es válida sobre los dominios clásicos (i.e., reales, enteros, Booleanos y los conjuntos) y nuevos dominios (posiblemente compuestos). En esta generalización hemos propuesto un esquema de ramificación para las restricciones de intervalo que extiende el esquema de propagación descrito en la Sección 3.5. Esta extensión da lugar a un esquema genérico para la resolución de restricciones de intervalo que permite la resolución de problemas formulados en términos de estas restricciones y definidos sobre cualquier conjunto de retículos (predefinidos o definidos por el usuario).

Hemos declarado las principales propiedades del nuevo esquema, tales como la terminación y la completitud. La terminación se garantiza mediante el mismo mecanismo que usamos para garantizar la terminación de nuestro esquema de propagación.

El esquema es totalmente transparente y es aplicable a sistemas cooperativos y que admiten multiples dominios.

## 5.7. Contribuciones

A pesar de que es bien conocido que, en muchos casos, el proceso de ramificación es clave para resolver completamente un CSP, la mayoría de la literatura enfocada a encontrar principios generales dentro de la comunidad de las restricciones y, en particular con respecto a los algoritmos de resolución, se concentra en el proceso de propagación (Apt, 1999; Fernández y Hill, 1999b; Van Hentenryck et al., 1992).

Las principales contribuciones de este capítulo son las siguientes:

- Primero, con respecto a la comunidad de las restricciones, nuestra propuesta es un intento de encontrar principios generales para el proceso de ramificación en la resolución de las restricciones de intervalo.

  Nuestro esquema puede ser usado para explicar el comportamiento operacional de muchos sistemas existentes tales como clp(FD) (Codognet y Diaz, 1996a), clp(B) (Codognet y Diaz, 1994), DecLic (Goualard et al., 1999), clp(B/FD) (Codognet y Diaz, 1996b), CLIP (Hickey, 2000), Conjunto (Gervet, 1997) o CLP(BNR) (Benhamou y Older, 1997).

- Segundo, este capítulo complementa el esquema genérico y cooperativo para la propagación de restricciones de intervalo propuesto en el Capítulo 3 de manera que el esquema resultante permite la resolución (completa) de la restricciones (en realidad de los problemas formulados en términos de las mismas). El esquema global es genérico, cooperativo y transparente. A nuestro entender sólo el sistema CHR cumple estas características. Ya que nuestro enfoque es totalmente diferente a ese de CHR, nuestro esquema es una alternativa a este sistema.

Como contribuciones menores, este capítulo también

- propone un nuevo mecanismo para normalizar las heurísticas de ordenación de variables. Por ello este mecanismo puede ser muy útil en los sistemas que admiten diversos dominios tales como los sistemas cooperativos;

- demuestra que, en ciertos casos, la forma de resolver un CSP depende de la relación de orden definida sobre las funciones de coste.

# Parte IV

# Marco Práctico

# Capítulo 6

# Un lenguaje de CLP Genérico, Colaborativo y de 2D

*The limits of my language*
*mean the limits of my world.*

Tractatus Logico-Philosophicus (1922)
Ludwig Wittgenstein 1889-1951

## 6.1. Motivación

Para demostrar que nuestras ideas, descritas en los capítulos anteriores, son viables, hemos implementado un lenguaje lógico para la resolución de restricciones de intervalo formuladas sobre un conjunto de retículos y cuyos fundamentos están basados en esos descritos en la Parte III.

En este capítulo proponemos un nuevo lenguaje para CLP llamado $clp(\mathcal{L})$ (i.e., CLP sobre conjuntos de retículos). El lenguaje $clp(\mathcal{L})$ es una propuesta para un resolutor flexible que permite la generación de nuevas restricciones definidas por el usuario (lo que significa un enfoque transparente sobre las restricciones) y la creación de nuevos dominios de computación también a nivel de usuario (lo que significa un enfoque transparente sobre los dominios de computación). Estos dominios podrían tener estructuras diversas pudiendo ser finitos, infinitos, discretos o continuos. El lenguaje también posibilita la interacción de los dominios de computación. En general decimos que $clp(\mathcal{L})$ proporciona un enfoque transparente de dos dimensiones (2D): sobre las restricciones y sobre los dominios (véase la Figura 1.3).

En este capítulo se describen las principales características del lenguaje $clp(\mathcal{L})$, se presenta brevemente una implementación prototipo y se muestran varios ejemplos que muestran la flexibilidad del lenguaje. Para más información ver (Fernández, 2000).

Debido a que este documento es un resumen del contenido global de la tesis y puesto que los fundamentos teóricos ya han sido presentados en los capítulos anteriores, a continuación sólo indicaremos las conclusiones obtenidas en este capítulo y las contribuciones del mismo.

## 6.2. Conclusiones

En este capítulo se introduce el lenguaje $clp(\mathcal{L})$ language y se muestran las principales características de una primera implementación prototipo.

EL lenguaje $clp(\mathcal{L})$ proporciona soporte para la resolución de restricciones de intervalo formuladas sobre cualquier conjunto de retículos sin restricción en su naturaleza. Por tanto, puede ser usado sobre los dominios usuale y sobre otros que el usuario defina para aplicaciones más específicas. La fundamentos básicos del lenguaje se basan en los principios de la teoría de retículos por lo que es fácil combinar los dominios de computación a partir de operadores de combinación de retículos. Puesto que los operadores pueden ser definidos sobre más de un dominio, la información puede transmitirse entre todos y cada uno de los dominios de computación.

En este capítulo se especifica como se pueden generar nuevos dominios desde la nada o a partir de la combinación de otros ya existentes. Se muestra además cómo definir restricciones de alto nivel genéricas o sobrecargadas y se muestran ejemplos de la utilidad de éstas.

También se han desarrollado minuciosamente una serie de ejemplos con el objetivo de mostrar la flexibilidad, transparencia, generalidad, cooperativismo y declarativid del lenguaje $clp(\mathcal{L})$.

En ningún momento se ha discutido la eficiencia del sistema puesto que el prototipo está construido usando las CHRs, las cuales a pesar de ser muy expresivas son altamente ineficientes como se demostró en el Capítulo 2. Sin embargo, está probado que el enfoque indexical es muy eficiente sobre el dominio finito, el dominio Booleano y sobre los reales (Codognet y Diaz, 1996a; Goualard et al., 1999). Puesto que el lenguaje $clp(\mathcal{L})$ está fuertemente basado en este enfoque, entonces si adaptamos las técnicas usadas para la implementación de sistemas tales como clp(FD), clp(B) y *DecLic* (los cuales siguen el enfoque de indexicals) a nuestro sistema, podemos vaticinar que la implementación resultante tendrá un rendimiento competitivo comparado con otros sistemas de CLP específicamente diseñados para dominios determinados.

Actualmente el esquema de ramificación de las restricciones de intervalo descrito en el Capítulo 5, no ha sido todavía implementado en el prototipo. Esto forma parte del trabajo futuro a realizar.

## 6.3. Contribuciones

Las principales contribuciones del capítulo son enumeradas a continuación:

- Primero, demuestra que las ideas teóricas descritas en la Parte III son factibles..

- Segundo, demuestra que un único sistema puede combinar un enfoque transparente sobre los dominios y las restricciones, un enfoque genérico y un enfoque cooperativo.
- Finalmente, propone un lenguaje nuevo que combina otras características potencialmente interesantes tales como la posibilidad de definir restricciones genéricas o sobrecargadas.

# Capítulo 7

# Conclusiones

*Roma locuta est; causa finita est*
*Roma has spoken; the case is concluded*

Traditional summary of words found in
Sermons (Antwerp, 1702), n.131, seat 10
St. Augustine of Hipo, Ad. 354-430

## 7.1. Resumen de los Resultados

En esta tesis, hemos propuesto un esquema para la resolución de restricciones de intervalo que combina tres características deseables: (1) generalidad real, (2) cooperativismo de los resolutores y (3) transparencia total en la definición de restricciones, dominios y el mecanismo de propagación. Además vaticina una cuarta propiedad importante: eficiencia en la resolución de las restricciones. El esquema ha sido desarrollado en cuatro etapas.

Primeramente, nosotros hemos comparado los diferentes enfoques transparentes (sobre las restricciones) que existen sobre el FD. La razón argumentada para elegir este dominio fue que los principales sistemas transparentes actualmente disponibles han sido implementados para este dominio. A partir de esta comparativa, hemos observado que los sistemas transparentes proporcionan una flexibilidad muy superior a los sistemas opacos, y ello sin una pérdida significativa de eficiencia. Como resultado de esta comparación, hemos elegido el modelo de *indexicals* como la base sobre la cual construir los fundamentos teóricos de nuestro esquema. La elección se justificó en el hecho de que este modelo mostró un rendimiento aceptable y una flexibilidad elevada en la formulación de problemas (discretos). Además el modelo fue catalogado como transparente.

Después hemos generalizado el modelo de indexicals desde el FD a cualquier dominio con estructura de retículo. El resultado es un esquema genérico para la propagación

de restricciones de intervalo que es válido para cualquier retículo, independiente de la cardinalidad de éste. Hemos mostrado que el esquema es útil para la mayoría de los dominios clásicos y que, además, es válido para nuevos dominios que son generados a partir de constructores y combinadores de retículos. Las principales propiedades del esquema, tales como la corrección, han sido demostradas. Para asegurar la terminación de la propagación usamos una función *precision/1* que es definida para cada dominio de computación. Como resultado añadido, la monotonía de las restricciones está asegurada por la misma formalización del esquema. Una consecuencia directa es que las restricciones anti-monótonas pueden ser detectadas fácilmente *a priori* (i.e., antes de su resolución) mediante un simple test que compruebe la validez de las restricciones en el marco teórico. Más aún, al igual que el modelo de indexicals para el FD, nuestro esquema para la propagación se basa en una restricción primitiva simple $x \sqsubseteq r$ que proporciona la especificación de la restricción y cierto control sobre su mecanismo de propagación .

La restricción primitiva puede estar definida a partir de operadores de restricción definidos sobre multiples dominios lo que da lugar a una nueva forma de cooperación en la que se permite que la información fluya entre los diferentes dominios soportados por el sistema (i.e., cualquier retículo). El concepto de restricción de alto nivel usado en CLP(FD) ha sido generalizado en nuestro esquema. Esta generalización incrementa el potencial de la cooperación y además posibilita la definición de restricciones genéricas o sobrecargadas. Obsérvese, además, que nuestro esquema cooperativo garantiza la *seguridad de tipos*. De hecho, los operadores de restricción se comportan como el interfaz de comunicación entre los resolutores y a partir de los cuales la información se propaga. Como ya se indicó anteriormente, las restricciones no-válidas pueden ser detectadas *a priori* y, por ello, todas las restricciones cooperantes (i.e., ésas que posibilitan la cooperación) admitidas en nuestro esquema son válidas.

Posteriormente, hemos completado el esquema para la resolución de restricciones de intervalo. El nuevo esquema es transparente, colaborativo y genérico y, por tanto, puede usarse tanto para la resolución clásica de CSPs como para la resolución parcial de los mismos. El esquema resultante, llamado el esquema de ramificación o de bifurcación, puede usarse sobre cualquier conjunto de retículos y es una generalización del conocido método 'split-and-solve' del lenguaje CLP(BNR), el cual resuelve restricciones de intervalo sobre los reales, los enteros y los Booleanos. El esquema de bifurcación está parametrizado en varios procedimientos que son parcialmente especificados (i.e., están declarados formalmente y las precondiciones y las poscondiciones definidas). Además, hemos estudiado algunas propiedades que son satisfechas por cualquier instancia del esquema, siempre que los procedimientos parcialmente especificados mantengan ciertas condiciones (que, por supuesto, también han sido especificadas). De nuevo, la terminación es asegurada mediante la función *precisión/1* que fue definida para el esquema de propagación.

Para demostrar que nuestras ideas son factibles, hemos definido un lenguaje para CLP basado en el marco teórico y, además, hemos implementado un prototipo para dicho lenguaje y mostrado un conjunto de ejemplos que ilustran la declaratividad del sistema.

## 7.2. Resumen de las Principales Contribuciones

Esta tesis ha tratado aspectos muy diferentes relacionados con la CLP. En cada capítulo hemos reseñado las principales contribuciones aportadas en el capítulo. Ahora resumimos las principales aportaciones de la tesis.

- Hemos comparado la eficiencia y algunos aspectos claves de la expresividad de ocho sistemas para CLP sobre el dominio finito. Hasta lo que conocemos, ésta es la primera vez que un número tan elevado de sistemas para CLP es comparado en el entorno de la comunidad de las restricciones. Hemos discutido los principales aspectos que deben ser tenidos en cuenta a la hora de escoger un sistema para resolver una aplicación específica. Además hemos proporcionado unas líneas directivas para escoger uno u otro sistema dependiendo de la aplicación a resolver y de los requisitos a conseguir (i.e., tiempo, declaratividad de la solución, etc.).

- Hemos propuesto una generalización del modelo de indexicals para el dominio finito a cualquier dominio con estructura de retículo, independientemente de la cardinalidad y naturaleza del dominio. Este modelo genérico es un esquema para la resolución de restricciones de intervalo. Se han probado las principales propiedades del modelo y se ha demostrado que son satisfechas por todas y cada una de sus instancias (incluso de esas cooperativas). La propuesta da lugar a un sistema transparente para las restricciones y para los dominios que, también, permite conectar resolutores definidos sobre dominios distintos. Un resultado inmediato de nuestra propuesta es que constituye una alternativa al lenguaje CHR. Además, al estar basado en el modelo de indexicals (el cual se ha probado que es bastante eficiente), podemos vaticinar que nuestro esquema garantiza también una eficiencia más que aceptable de cualquier sistema basado en él.

- Hemos demostrado que la cooperación de resolutores puede ser integrada en un sistema único bajo un enfoque transparente.

## 7.3. Trabajo Futuro

A pesar de que se han alcanzado todos los objetivos trazados en el Capítulo 1, esta tesis abre algunas posibilidades que complementan el trabajo aquí realizado. A continuación, se describen algunas de las líneas de investigación abiertas.

Con respecto a nuestra comparación de sistemas de restricciones sobre el dominio finito, queremos resaltar que hemos comparado las librerías sobre este dominio y no los sistemas globales. Estudios similares deberían hacerse sobre otros dominios interesantes tales como el dominio de intervalos (Benhamou, 1995) o el dominio real (Jaffar et al., 1992). También deberían considerarse enfoques alternativos tales como la programación lineal.

En nuestro marco teórico, hemos impuesto que el conjunto de variables restringidas para cada dominio de computación tiene que ser disjunto. Sin embargo, debería

ser posible eliminar esta restricción y considerar sub-retículos de los retículos, como dominios de computación. Esto permitiría tener conjuntos de variables restringidas en un sub-retículo que serían además restringidas en el retículo del que forman parte.

Hemos estudiado las propiedades genéricas del esquema de ramificación, y hemos impuesto algunas condiciones sobre las heurísticas de ordenación de variables y valores. Sin embargo, estas condiciones no garantizan que el espacio de búsqueda es mínimo. En otras palabras, no se garantiza que exista información redundante en el árbol de búsqueda. Por lo tanto, es interesante estudiar si existen propiedades que garanticen la *minimalidad* del espacio de búsqueda y, si existen, declararlas y demostrar que, efectivamente, aseguran la minimalidad. Por ejemplo, sobre las funciones de partición (ver Definición 5.10) podríamos recomendar que deberían satisfacer la siguiente propiedad:

$$\forall i, j \in \{1, \ldots, k\}, i \neq j : glb_{\mathcal{C}_L^X}(c_i, c_j) \text{ es inconsistente}$$

la cual asegura (en teoría) que "todas las particiones son disjuntas", y por tanto, la redundancia de información es evitada.

Observa también que en el esquema $resuelve_{\alpha+}/3$ no se ha tenido en cuenta la eficiencia, y sólo se ha considerado la función de coste y la relación de orden en su rango como instrumentos para encontrar una solución óptima. Evidentemente, esto es claramente ineficiente puesto que el árbol de búsqueda tiene que ser atravesado completamente. Naturalmente nuestro objetivo fue estudiar los principios generales del proceso de la resolución de restricciones de intervalo en los problemas de optimización. Aún así, sería interesante investigar si nuestro esquema puede ser reformulado con la idea de realizar podas del árbol del búsqueda en los casos en los que se conoce que no habrá una solución mejor, que la actualmente encontrada, en el subárbol actual.

La implementación descrita en esta tesis se ha construido usando CHRs, las cuales proporcionan una expresividad muy alta pero son muy ineficientes. Por lo tanto, en un futuro a corto plazo, la idea es implementar nuestro esquema en una arquitectura más eficiente. Varios enfoques se están considerando actualmente (Diaz y Codognet, 1993).

También estamos actualmente estudiando la integración de nuestro esquema en un lenguaje lógico funcional. Estos lenguajes combinan propiedades lógicas con características funcionales lo que los convierten en entornos adecuados para la implantación de nuestro sistema genérico. Creemos que existe una fuerte analogía de los conceptos de *tipo* y *función* en los lenguajes lógico funcionales con respecto a los conceptos de *dominio* y *operadores de restricción* en nuestro marco teórico. Por ello, pensamos que nuestro esquema podría integrarse bastante bien en este tipo de lenguajes. En este sentido ya hemos realizado una primera aproximación (Cazorla, 2001).

# Bibliografía


Abdennadher, S. (1997). Operational semantics and confluence of constraint propagation rules. En Smolka, G. (ed.), *3rd International Conference on Principles and Practice of Constraint Programming (CP'97)*, núm. 1330 de LNCS, pp. 252–266, Linz, Austria. Springer-Verlag.

Abdennadher, S., Frühwirth, T., y Meuss, H. (1999). Confluence and semantics of constraint simplification rules. *Constraints*, 4(2):133–165.

Aggoun, A., Chan, D., Dufresne, P., Falvey, E., Grant, H., Herold, A., Macartney, G., Meier, M., Miller, D., Mudambi, S., Perez, B., Rossum, E. V., Schimpf, J., Periklis, Tsahageas, A., y de Villeneuve, D. (1995). $ECL^iPS^e$ 3.5, User manual. European Computer -Industry Research Centre (ECRC). Munich.

Aït-kaci, H. (1999). *Warren's Abstract Machine: a tutorial reconstruction*. The MIT Press, Cambridge, MA.

Apt, K. (1999). The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2):179–210.

Barth, P. y Bockmayr, A. (1996). Modelling 0-1 problems in CLP($\mathcal{PB}$). En Wallace, M. (ed.), *2nd International Conference on the Practical Application of Constraint Technology (PACT'96)*, pp. 1–9, London, UK. Prolog Management Group.

Benhamou, F. (1995). Interval constraint logic programming. En Podelski, A. (ed.), *Constraint Programming: Basics and Trends*, núm. 910 de LNCS, pp. 1–21, Châtillon-sur-Seine, France. Springer-Verlag.

Benhamou, F. y Older, W. (1997). Applying interval arithmetic to real, integer and Boolean Constraints. *The Journal of Logic Programming*, 32(1):1–24.

Carlson, B., Carlsson, M., y Diaz, D. (1994). Entailment of finite domain constraints. En Van Hentenryck, P. (ed.), *11th International Conference on Logic Programming (ICLP'94)*, pp. 339–353, Santa Margherita Ligure, Italy. The MIT Press.

Carlsson, M., Ottosson, G., y Carlson, B. (1997). An open-ended finite domain constraint solver. En Montanari, U. y Rossi, F. (eds.), *9th International Symposium on Programming Languages: Implementations, Logics and Programs (PLILP'97)*, núm. 1292 de LNCS, pp. 191–206, Southampton, UK. Springer-Verlag.

Cazorla, J. (2001). Programación lógico-funcional con restricciones genéricas. Tesis de licenciatura, ETSI (Informática) Málaga University, Málaga, Spain. Directed by Antonio J. Fernández (in spanish).

Codognet, P. y Diaz, D. (1994). clp(B): combining simplicity and efficiency in Boolean constraint solving. En *6th International Symposium on Programming Languages Implementation and Logic Programming (PLILP'94)*, núm. 844 de LNCS, pp. 244–260, Madrid, Spain. Springer-Verlag.

Codognet, P. y Diaz, D. (1996a). Compiling constraints in clp(FD). *The Journal of Logic Programming*, 27(3):185–226.

Codognet, P. y Diaz, D. (1996b). A simple and efficient boolean solver for constraint logic programming. *The Journal of Automated Reasoning*, 17(1):97–129.

Cras, J.-Y. (1993). A review of industrial constraint solving tools. *AI Intelligence*.

Csontó, J. y Paralič, J. (1997). A look at CLP: theory and application. *Applied Artificial Intelligence*, 11:59–69.

Davey, B. y Priestley, H. (1990). *Introduction to lattices and order*. Cambridge University Press, Cambridge, England.

Diaz, D. (1994). clp(FD) 2.21 User's manual. INRIA-Rocquencourt, France.

Diaz, D. (1995). *Etude de la compilation des langages logiques de programmation par contraintes sur les domaines finis: le système clp(FD)*. Tesis Doctoral, l'université d'Orléans.

Diaz, D. y Codognet, P. (1993). A minimal extension of the WAM for clp(FD). En Warren, D. (ed.), *10th International Conference on Logic Programming (ICLP'93)*, pp. 774–790, Budapest, Hungary. The MIT Press.

Fernández, A. (1997). Towards a glass-box typed CLP language. En Fisher, M. (ed.), *Workshop on Automated Reasoning (WAR'97)*, AISB Workshop and Tutorial Series, pp. 7–8, Manchester. University of Manchester.

Fernández, A. (2000). clp(L) version 0.21, user manual. Available at *http://www.lcc.uma.es/∼afdez/generic*.

Fernández, A. y Hill, P. (1997a). Boolean and finite domain solvers compared using self referential quizzes. En Falaschi, M., Navarro, M., y Policriti, A. (eds.), *Joint Conference on Declarative Programming (APPIA-GULP-PRODE'97)*, pp. 533–544, Grado, Italy. CLEUP. Also available as Technical Report ref.97.03, School of Computer Studies, University of Leeds, January, 1997.

Fernández, A. y Hill, P. (1997b). Finite domain solvers compared using self referential quizzes. En *Sixièmes Journées Francophones de Programmation Logique et*

*Programmation par Contraintes (JFPLC'97)*, Orléans, France. LIFO. (Poster Session). Rapport de Recherche.

Fernández, A. y Hill, P. (1998a). A design for a generic constraint solver for ordered domains. En *Types for Constraint Logic Programming (TCLP'98)*, Manchester.

Fernández, A. y Hill, P. (1998b). A generic execution model for CLP(X). Research Report LCC-ITI 98/16, Universidad de Málaga, Departamento de Lenguajes y Ciencias de la Computación.

Fernández, A. y Hill, P. (1998c). An impartial efficiency comparison of FD constraints systems. En Maher, M. y Puget, J.-F. (eds.), *4th International Conference on Principles and Practice of Constraint Programming (CP'98)*, núm. 1520 de LNCS, pp. 468, Pisa, Italy. Springer-Verlag. Also available as Technical Report ref.98.18, School of Computer Studies, University of Leeds, September, 1998.

Fernández, A. y Hill, P. (1999a). Constrant solving on lattices. En Meo, C. y Vilares, M. (eds.), *International Joint Conference on Declarative Programming (APPIA-GULP-PRODE'99)*, pp. 105–120, L'Aquila, Italy. Gruppo Tipografico Editoriale.

Fernández, A. y Hill, P. (1999b). An interval lattice-based constraint solving framework for lattices. En Middeldorp, A. y Sato, T. (eds.), *4th International Symposium on Functional and Logic Programming (FLOPS'99)*, núm. 1722 de LNCS, pp. 194–208, Tsukuba, Japan. Springer-Verlag.

Fernández, A. y Hill, P. (2000a). A comparative study of eight constraint programming languages over the Boolean and finite domains. *Constraints*, 5(3):275–301.

Fernández, A. y Hill, P. (2000b). Constraint propagation on multiple domains. En Alpuente, M. (ed.), *9th International Workshop on Functional and Logic Programming (WFLP'00)*, pp. 455–469, Benicássim, Spain. Universidad Politécnica de Valencia.

Fernández, A. y Hill, P. (2001a). Branching: the essence of constraint solving. En Apt, K., Barták, R., Monfroy, E., y Rossi, F. (eds.), *ERCIM Workshop on Constraints*, Prague, Czech Republic. Charles University/Faculty of Mathematics and Physics.

Fernández, A. y Hill, P. (2001b). A constraint system for lattice (interval) domains. *ACM Transactions on Programming Languages and Systems (TOPLAS)*. Submitted December 2000; subjected to revision in November 2001.

Frühwirth, T. (1994). Constraint handling rules. En Podelski, A. (ed.), *Constraint Programming: Basics and Trends*, núm. 910 de LNCS, pp. 90–107, Châtillon-sur-Seine, France. Springer-Verlag.

Frühwirth, T. (1998). Theory and practice of constraint handling rules. *The Journal of Logic Programming*, 37:95–138.

Frühwirth, T. (1999). Compiling constraint handling rules into Prolog with attributed variables. En Nadathur, G. (ed.), *International Conference on Principles and Practice of Declarative Programming (PPDP'99)*, núm. 1702 de LNCS, pp. 117–133, Paris, France. Springer-Verlag.

G., S. y Treinen R., e. (1995). DFKI Oz documentation series. German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany.

Gervet, C. (1997). Interval propagation to reason about sets: definition and implementation of a practical language. *Constraints*, 1(3):191–244.

Goualard, F., Benhamou, F., y Granvilliers, L. (1999). An extension of the WAM for hybrid interval solvers. *The Journal of Functional and Logic Programming*, 1999(1):1–36. Special issue of Workshop on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages.

Haralick, R. y Elliot, G. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313.

Henz, M. (1996). Don't be puzzled! En *Workshop on Constraint Programming in conjunction with the 2nd International Conference on Principles and Practice of Constraint Programming (CP'96)*, Cambridge, Massachusetts, USA.

Hickey, T. (2000). CLIP: a CLP(Intervals) dialect for metalevel constraint solving. En Pontelli, E. y Costa, V. (eds.), *2nd International Workshop on Practical Aspects of Declarative Languages (PADL'2000)*, núm. 1753 de LNCS, pp. 200–214, Boston, USA. Springer-Verlag.

If/Prolog (1994). *IF/Prolog V5.0A, constraints package.* Siemens Nixdorf Informationssysteme AG, Munich, Germany.

Ilog (1995). Ilog SOLVER, reference manual, version 3.1.

Jaffar, J. y Lassez, J. (1987). Constraint logic programming. En *14th ACM Symposium on Principles of Programming Languages (POPL'87)*, pp. 111–119, Munich, Germany. ACM Press.

Jaffar, J. y Maher, M. (1994). Constraint logic programming: a survey. *The Journal of Logic Programming*, 19-20:503–581.

Jaffar, J., Michaylov, S., Stuckey, P., y Yap, R. (1992). The CLP($\Re$) language and system. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395.

Mackworth, A. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8:99–118.

Müller, T. y Müller, M. (1997). Finite set constraints in Oz. En Bry, F., Freitag, B., y Seipel, D. (eds.), *13th Workshop on Logic Programming*, pp. 104–115, München. Technische Universität.

Müller, T. y Würtz, J. (1996). Interfacing propagators with a concurrent constraint language. En *JICSLP96 Post-conference workshop and Compulog Net Meeting on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages*, pp. 195–206, Bonn, Germany.

PAPPACT'98 (1998). *6th International Conference of Practical Application of Prolog and the 4th International Conference on the Practical Application of Constraint Technology*. Publisher Practical Application Company Ltd, London, UK.

Puget, J.-F. y Leconte, M. (1995). Beyond the glass box: constraints as objects. En J. W. Lloyd (ed.), *International Symposium on Logic Programming (ILPS'95)*, pp. 513–527, Portland, Oregon. The MIT Press.

Refalo, P. y Van Hentenryck, P. (1996). CLP($\Re_{lin}$) revised. En Maher, M. (ed.), *Joint International Conference and Symposium on Logic Programming (JICSLP'96)*, pp. 22–36, Bonn, Germany. The MIT Press.

Sicstus manual (1994). *SICStus Prolog user's manual, release* 3#5. By the Intelligent Systems Laboratory, Swedish Institute of Computer Science.

Sidebottom, G. (1993). *A language for optimizing constraint propagation*. Tesis Doctoral, Simon Fraser University, Burnaby, Canada.

Sidebottom, G. y Havens, W. (1992). Hierarchical arc consistency for disjoint real intervals in constraint logic programming. *Computational Intelligence*, 8(4):601–623.

Smolka, G. (1995). The Oz programming model. En Van Leeuwen, J. (ed.), *Computer Science Today*, núm. 1000 de LNCS, pp. 324–343, Berlin. Springer-Verlag.

Van Hentenryck, P. (1989). *Constraint satisfaction in logic programming*. The MIT Press, Cambridge, MA.

Van Hentenryck, P., Deville, Y., y Teng, C. (1992). A Generic Arc-Consistency Algorithm and its Specializations. *Artificial Intelligence*, 57(2-3):291–321.

Van Hentenryck, P., Saraswat, V., y Deville, Y. (1991). Constraint processing in cc(FD). Unpublished draft.

Walinsky, C. (1989). CLP($\Sigma^*$): constraint logic programming with regular sets. En Levi, G. y Martelli, M. (eds.), *6th International Conference on Logic Programming (ICLP'89)*, pp. 181–196, Lisbon, Portugal. The MIT Press.

Zhou, N.-F. (1997). B-Prolog user's manual (version 2.1). Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Fukuoka, Japan.