Designing Constraint Maintainers for User Interaction

Lambert Meertens[*]

Department of Algorithmics and Architecture, CWI, Amsterdam, and

Department of Computing Science, Utrecht University, The Netherlands

`www.cwi.nl/~lambert`

Printed June 23, 1998

[*]Work performed while visiting Kestrel Institute, Palo Alto.

0 Introduction

Constraints for user interfaces.. Constraints are used in the construction of user interfaces in two rather different ways. Both ways have in common that they afford a certain amount of Direct Manipulation, but to different classes of users.

One use of constraints is in the graphical aspects of user-interface management systems, comprising toolboxes for specifying the appearance of the visual manifestation of the user interface on the screen, including such things as the position of various widgets in a window [2, 7]. The DM capability here is provided to the interface designers.

Another use of constraints is to specify the relationship between *views* (visually presented objects) and the object values they represent. The simplest case of this is known as the *Model-View-Controller* paradigm [5]. More generally, there is a network of "documents", comprising views as well as other objects linked together by constraints. This formed the basis of Higgens [3, 4], in which the user interface was generated "off-line", and was the pervading architecture of the Views system [6, 8]. Here, it is the end-user who uses "Direct Manipulation" (in an extended sense) to control the applications(s). For example, a "folder" or "workspace", that is, a collection of documents, might have a view consisting of (possibly embellished) presentations of the names of these documents. The constraint linking these two is: names-of-documents-in-folder = names-in-view. If a document in the folder is renamed by any means, its new name then appears in the view. Conversely, if a name is edited and changed in the view, the corresponding document is renamed. The constraints are like physical laws which — if designed in a reasonable way — impart predictable behaviour to the system (compare ThingLab [1]). Such predictability is essential for usability.

The evolution of user-interface design has been going for quite some time now in the direction of direct user control, coupling the visual manifestation to the system functionality in a way that is readily expressed in terms of constraints. Preferably, the "constraint architecture" of a system is reflected in its software architecture. The alternative is that the constraint maintenance is programmed-out *ad hoc*, which — although cumbersome and error prone — is feasible in systems of modest and fixed functionality. In either case, the quality and consistency of the functional design of software may benefit from using a constraint-based formulation for its specification.

Constraint maintenance. We focus on networks that have no cycles and in which each constraint is a two-way constraint, linking two objects at a time. The simplest non-trivial case is a network of two objects and one constraint, as depicted in Figure 1. By some external process (for example,

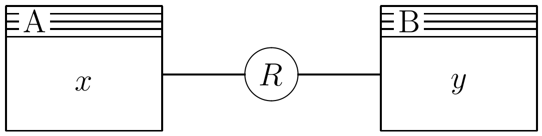

Figure 1: A simple network of two objects A and B linked by the constraint that their respective contents x and y be such that xRy holds.

the action of a user editing an object) the contents of an object may be changed. If (referring to the situation in Figure 1) the value of A gets changed to x', and thereby the constraint is violated, that is, $x'Ry$ does not hold, constraint maintenance intercedes and replaces the contents of B by some value y' such that $x'Ry'$ holds. Conversely, if B is changed, the continual process of constraint maintenance may need to change A in order to repair the constraint. Suppose now, for concreteness, that A and B are numeric objects, and that R is the relation "$\leq$" ("is at most"). Assume that $x = 2.5$ and $y = 3.25$. If the user would change x to $x' = 1$, the constraint is not violated and B does not have to be updated. If, on the other hand, the user changes x to $x' = 4.0$, y has to be replaced by some value y' such that $4.0 \leq y'$. Now there are many such values, for example 5, 15.154 and 100000. Which to choose? The claim is that although there are many formal solutions to the constraint, some are *more intuitively natural* in terms of user expectation than others. In particular, in this example, the intuitively natural choice is $y' = 4.0$. A simple physical system embodying this constraint is shown in Figure 2. Generalizing the example leads to the *Principle of Least Change*:

The action taken by the maintainer of a constraint after a violation should change no more than is needed to restore the constraint.

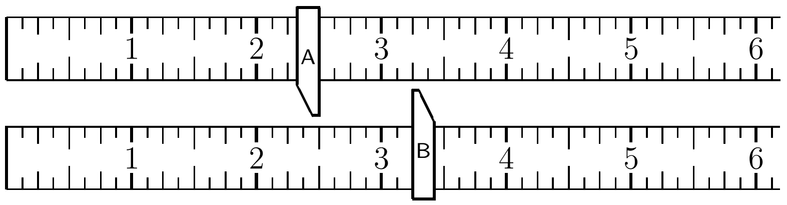

Figure 2: A physical embodiment of the constraint "$\leq$". If slider A is moved from position 2.5 to position 4.0, it will engage with slider B and force it to also move to position 4.0.

Constraint maintainers. A *constraint maintainer* is a pair of functions that can be used to compute the necessary updates for constraint maintenance. These functions have two arguments: the new value at one end of the link, and the old value at the other end of the link. They will usually be denoted by the infix binary operators $\lhd$ and $\rhd$. If the relationship xRy holds at some time, and there is a change $y \leadsto y'$, then $x \lhd y'$ is computed, and the update $x \leadsto x \lhd y'$ is effected. Conversely, if relationship xRy holds at some time, and there is a change $x \leadsto x'$, then $x' \rhd y$ is computed, and the update $y \leadsto x' \rhd y$ is effected. This paper is concerned with the question: How to design, given a constraint R, constraint maintainers of it that are "intuitively natural".

1 Basic definitions and notation

Quantifications. *Quantifications* are written as in $\exists(n : n \in \mathbb{N} : 3^n + 4^n = 5^n)$. The quantifier, in this case existential, is all in front. Preceding the first colon we find the dummy variables, between the two colons their range, and following the second colon a proposition, typically depending on the dummy variables. The range is often omitted, meaning that the range of each dummy is restricted by the typings given in the context for the functions applied to it. So if $f\colon \mathbb{N} \to A$, the statement $\forall(n :: f(n+1) > f(n))$ means: $\forall(n : n \in \mathbb{N} : f(n+1) > f(n))$.

Function comprehensions are also written in quantifier notation, but without the quantifier. The notation $(n : n \in \mathbb{N} : 2^n)$ means: the function mapping a natural number n to 2^n. A more traditional notation for this function is $\lambda n \in \mathbb{N} \bullet 2^n$. We occasionally use a dummy "hole" (a small square $\square$) to get a shorthand for a one-argument function comprehension. For example, "$\square + 1$" stands for "$(x :: x + 1)$".

A proposition between *square brackets* is an abbreviation for universal quantification; it means: that proposition universally quantified over all free variables of the proposition that are not bound in its context. For example,

"If x is positive, then $[y \geq x \Rightarrow y > 0]$"

means

"If x is positive, then $\forall(y :: y \geq x \Rightarrow y > 0)$".

The same convention is used in definitions, as in this definition of function $\sqrt{\ }$ on the non-negative real numbers: $[y = \sqrt{x} \;\equiv\; y \geq 0 \wedge y^2 = x]$.

Pairs. If $x \in A$ and $y \in B$, the pair $\langle x, y\rangle$ denotes the corresponding element of $A \times B$. The projections are denoted by π_L and π_R, with

$$\begin{array}{ll} \pi_L\colon A \times B \to A & \pi_R\colon A \times B \to B \\ [\,\pi_L\langle x, y\rangle \;=\; x\,] & [\,\pi_L\langle x, y\rangle \;=\; y\,] \end{array}$$

(Normally, we denote function application with parentheses around the argument, as in "$f(x)$". If, however, the argument is a pair, as above, or a set comprehension, as in "$f\{x, y, x\}$", the parentheses are omitted.)

Converse and sections. Given a function $\oplus\colon A \times B \to C$, its *converse* $\breve{\oplus}\colon B \times A \to C$ is defined by (using infix notation) : $[y \,\breve{\oplus}\, x = x \oplus y]$. Converse is an involution: $\breve{\breve{\oplus}} = \oplus$.

Furthermore, for fixed $x \in A$, the *left-section* of $\oplus$ for x, denoted by $x\oplus$, is the function $x \oplus \square\colon B \to C$. Likewise, for fixed $y \in B$, the *right-section* of $\oplus$ for y, denoted by $\oplus y$, is the function $\square \oplus y\colon A \to C$. So $x\oplus = (y :: x \oplus y)$ and $\oplus y = (x :: x \oplus y)$. Note that $x\oplus = \breve{\oplus} x$.

Predicates and sets. A *predicate* is a truth-valued function defined on some domain.

We *identify* predicates and sets. In particular, a predicate p is identified with the set of all values satisfying p. So $[x \in p \;\equiv\; p(x)]$. Thus, the predicate $(n : n \in \mathbb{N} : 2^n = n^2)$ also denotes the set $\{2, 4\}$. The size of a finite set s is denoted by $\#s$.

The set of all subsets of some set s will be denoted by $\mathcal{P}s$: $[t \in \mathcal{P}s \;\equiv\; t \subseteq s]$, while the set of all *non-empty* subsets of s is denoted by $\mathcal{P}_{\!+}s$. So $[\, t \in \mathcal{P}_{\!+}s \;\equiv\; t \in \mathcal{P}s \,\wedge\, t \neq \emptyset \,]$.

Relations. A (binary) *relation* is a predicate whose domain is some cartesian-product set $A \times B$. To indicate the typing of a relation R with that domain we write R: $A \sim B$. We use infix notation: for $x \in A$ and $y \in B$, rather than writing $R\langle x, y\rangle$ or $\langle x, y\rangle \in R$, we normally write xRy.

A relation R is called *ditotal* whenever both $[\exists(x :: xRy)\,]$ and $[\exists(y :: xRy)\,]$.

Given a relation R: $A \sim B$, we have for its *converse* $\breve{R}$: $B \sim A$ and $[y\breve{R}x \equiv xRy]$. If R is ditotal, then so is $\breve{R}$.

Given two relations R: $A \sim B$ and S: $B \sim C$, we can form their *composition* $(R\,;S)$: $A \sim C$, defined by: $[x(R\,;S)z \equiv \exists(y : y \in B : xRy \wedge ySz)\,]$. If R and S are ditotal, then so is $(R\,;S)$.

Given a function f: $A \to B$, its *relational embedding* is the relation $\langle\!\langle f\rangle\!\rangle$: $A \sim B$ defined by: $[x\langle\!\langle f\rangle\!\rangle y \;\equiv\; f(x) = y]$. The relation $\langle\!\langle f\rangle\!\rangle$ is ditotal whenever f is surjective. Embedding is an injective operation: $[\langle\!\langle f\rangle\!\rangle = \langle\!\langle g\rangle\!\rangle \;\equiv\; f = g]$. A relation that is the relational embedding of some function is called *functional*. Note the inversion in $[\,(\langle\!\langle f\rangle\!\rangle\,;\langle\!\langle g\rangle\!\rangle) = \langle\!\langle g\circ f\rangle\!\rangle\,]$.
Warning. Do not confuse $\langle\!\langle\breve{\oplus}\rangle\!\rangle$ with $\langle\!\langle\oplus\rangle\!\rangle\breve{}$ (that, is, $\breve{R}$ where $R = \langle\!\langle\oplus\rangle\!\rangle$). If $\oplus$: $A \times B \to C$, then $\langle\!\langle\breve{\oplus}\rangle\!\rangle$: $B \times A \,\sim\, C$, whereas $\langle\!\langle\oplus\rangle\!\rangle\breve{}$: $C \,\sim\, A \times B$.

2 Properties of maintainers

Given a constraint specified as a relation R: $A \sim B$, a maintainer of R is a pair of functions $\lhd$: $A \times B \to A$ and $\rhd$: $A \times B \to B$ that can be used to restore the validity of xRy after a change to x or y. The primary requirement on $\lhd$ is that the assignment $x{:=}x{\lhd}y$ establishes xRy, which is expressed by $[\, wp(x{:=}x{\lhd}y, xRy) \,]$. By wp-calculus this is equivalent to

$$[\, (x{\lhd}y)Ry \,]$$

Dually, for $\rhd$ the primary requirement is

$$[\, xR(x{\rhd}y) \,]$$

These requirements are all that is strictly necessary to guarantee that consistency can be restored after any autonomous change to an object in an acyclic network all of whose constraints are ditotal relations. However, we shall impose further requirements on maintainers, requirements that are justified from an ergonomic point of view. Later, we shall consider the requirement that restoring changes be as 'small' as possible. The notion of 'small' here is hard to formalize in a general way. In this section we consider only the weaker version of this requirement that no unnecessary changes be made. More precisely, if, after an assignment to y, the validity of xRy is not destroyed, then the effect of $x{:=}x{\lhd}y$ should be nil, the same as that of the skip $x{:=}x$. This amounts to

$$[\, xRy \Rightarrow x{\lhd}y = x \,]$$

Dually, we require

$$[\, xRy \Rightarrow x{\rhd}y = y \,]$$

Summing this up, we have

Definition (maintainer): A *maintainer* of a constraint

$$R\colon A \sim B$$

is a pair of functions

$$\begin{aligned} \lhd &: A \times B \to A \\ \rhd &: A \times B \to B \end{aligned}$$

satisfying the following four requirements:

$$\begin{aligned} &\lhd\text{-EST}: && [\,(x \lhd y) R y\,] \\ &\rhd\text{-EST}: && [\,x R (x \rhd y)\,] \\ &\lhd\text{-SKIP}: && [\,x R y \Rightarrow x \lhd y = x\,] \\ &\rhd\text{-SKIP}: && [\,x R y \Rightarrow x \rhd y = y\,] \end{aligned}$$

We also say that $\langle \lhd, \rhd \rangle$ *maintains* R.

Given a constraint R as above, functions $\lhd$ and $\rhd$ will be called *suitably typed* for R if the typing is as in this definition.

The requirements for $\lhd$ and for $\rhd$ are independent, giving rise to the notion of a *semi-maintainer*, being just one unidirectional half of a full (bidirectional) maintainer. In general a constraint can have several maintainers, and if both $\langle \lhd_0, \rhd_0 \rangle$ and $\langle \lhd_1, \rhd_1 \rangle$ maintain R, then so do $\langle \lhd_0, \rhd_1 \rangle$ and $\langle \lhd_1, \rhd_0 \rangle$.

Duality Principle: If the pair $\langle \lhd, \rhd \rangle$ maintains a constraint R, the pair $\langle \breve{\rhd}, \breve{\lhd} \rangle$ maintains $\breve{R}$.

Proof. Immediate by the definitions. *End of proof.*

Duality can be used for economy of proofs. Each proposition about maintainers has a dual, which is obtained from it by performing the substitution suggested by the Duality Principle. For example, the dual of $\lhd$-EST is $\rhd$-EST.

In general a constraint can have several maintainers, but for the important case of a functional relation one of the update functions is uniquely determined:

Theorem 1: If $\langle \lhd, \rhd \rangle$ maintains $\langle\!\langle f \rangle\!\rangle$, $[\,x \rhd y = f(x)\,]$.

Proof.

$$
\begin{array}{ll}
 & x \triangleright y = f(x) \\
\equiv & \quad \{ \text{ definition of } \langle\!\langle f \rangle\!\rangle \ \} \\
 & x \langle\!\langle f \rangle\!\rangle (x \triangleright y) \\
\equiv & \quad \{ \ \triangleright\text{-EST } \} \\
 & \mathsf{true}
\end{array}
$$

End of proof.

In general there is no easy way to obtain the other semi-maintainer. For example, f might be an encryption function in public-key cryptography. Finding $\triangleleft$ would amount to breaking that cryptosystem.

Theorem 2: Let $A \neq \emptyset$ and $B \neq \emptyset$. If constraint $R \colon A \sim B$ has a maintainer, it is ditotal.

Proof. Assume R has a maintainer. To show that R is ditotal, we must be able to find, given $x \in A$, a $y \in B$ such that xRy. So assume $x \in A$. Let η be any element of B, and take $y = x \triangleright \eta$. Then,

$$
\begin{array}{ll}
 & xRy \\
\equiv & \quad \{ \text{ definition of } y \ \} \\
 & xR(x \triangleright \eta) \\
\equiv & \quad \{ \ \triangleright\text{-EST } \} \\
 & \mathsf{true}
\end{array}
$$

Dually, we can find likewise, for any $y \in B$, a related $x \in A$. *End of proof.*

Later, in Theorem 9, we shall see that the converse holds as well: any ditotal relation has a maintainer.

Theorem 3: For a maintainer $\langle \lhd, \rhd \rangle$ of a constraint R each of the following holds.

$\lhd$-CHAR :	$[\, x \lhd y = x \equiv xRy \,]$
$\rhd$-CHAR :	$[\, x \rhd y = y \equiv xRy \,]$
$\lhd$-ADJ :	$[\, x \rhd y = y \Rightarrow x \lhd y = x \,]$
$\rhd$-ADJ :	$[\, x \lhd y = x \Rightarrow x \rhd y = y \,]$
ADJ :	$[\, x \lhd y = x \equiv x \rhd y = y \,]$
$\lhd\lhd$-SKIP :	$[\, (x \lhd y) \lhd y = x \lhd y \,]$
$\lhd\rhd$-SKIP :	$[\, (x \lhd y) \rhd y = y \,]$
$\rhd\rhd$-SKIP :	$[\, x \rhd (x \rhd y) = x \rhd y \,]$
$\rhd\lhd$-SKIP :	$[\, x \lhd (x \rhd y) = x \,]$

Proof. Let x and y be fixed throughout. For $\lhd$-CHAR the proof of the equivalence proceeds by cyclic implication:

$$
\begin{array}{ll}
 & x \lhd y = x \\
\Leftarrow & \quad \{\ \lhd\text{-SKIP}\ \} \\
 & xRy \\
\equiv & \quad \{\ \lhd\text{-EST}\ \} \\
 & (x \lhd y)Ry \equiv xRy \\
\Leftarrow & \quad \{\ \text{Leibniz}\ \} \\
 & x \lhd y = x
\end{array}
$$

By duality this also establishes $\rhd$-CHAR. Together these two give an easy proof for ADJ:

$$
\begin{array}{ll}
 & x \lhd y = x \\
\equiv & \quad \{\ \lhd\text{-CHAR}\ \} \\
 & xRy \\
\equiv & \quad \{\ \rhd\text{-CHAR}\ \} \\
 & x \rhd y = y
\end{array}
$$

The statements $\lhd$-ADJ and $\rhd$-ADJ are weaker versions that follow immediately. For $\lhd\lhd$-SKIP we calculate as follows:

$$\begin{array}{ll}
 & (x \lhd y) \lhd y = x \lhd y \\
\equiv & \quad \{\ \lhd\text{-CHAR, with } x \lhd y \text{ for } x\ \} \\
 & (x \lhd y) R y \\
\equiv & \quad \{\ \lhd\text{-EST}\ \} \\
 & \mathsf{true}
\end{array}$$

Similarly, for $\lhd\rhd$-SKIP:

$$\begin{array}{ll}
 & (x \lhd y) \rhd y = y \\
\equiv & \quad \{\ \rhd\text{-CHAR, with } x \lhd y \text{ for } x\ \} \\
 & (x \lhd y) R y \\
\equiv & \quad \{\ \lhd\text{-EST}\ \} \\
 & \mathsf{true}
\end{array}$$

The remaining two properties follow by duality. *End of proof.*

Conversely, any pair of functions having all these properties maintains R. To establish maintainership it suffices in fact to prove only a few of these properties, since not all are independent. In practice this may be an easier way than directly verifying the definition. Before going into this, we list a few properties that hold for any suitably typed pair of functions $\lhd$ and $\rhd$, whether a maintainer of R or not. They are useful in later proofs.

Lemma 4:

(a)	$\rhd\lhd$-SKIP	$\Rightarrow$	$\lhd$-ADJ
(b)	$\lhd\rhd$-SKIP	$\Rightarrow$	$\rhd$-ADJ
(c)	($\lhd$-ADJ $\wedge$ $\rhd$-ADJ)	$\equiv$	ADJ
(d)	$\lhd$-CHAR	$\Rightarrow$	$\lhd$-SKIP
(e)	$\rhd$-CHAR	$\Rightarrow$	$\rhd$-SKIP
(f)	ADJ	$\Rightarrow$	($\lhd$-CHAR $\equiv$ $\rhd$-CHAR)
(g)	$\lhd$-CHAR	$\Rightarrow$	($\lhd\lhd$-SKIP $\equiv$ $\lhd$-EST)
(h)	$\rhd$-CHAR	$\Rightarrow$	($\rhd\rhd$-SKIP $\equiv$ $\rhd$-EST)
(i)	$\rhd$-CHAR	$\Rightarrow$	($\lhd\rhd$-SKIP $\equiv$ $\lhd$-EST)
(j)	$\lhd$-CHAR	$\Rightarrow$	($\rhd\lhd$-SKIP $\equiv$ $\rhd$-EST)

Proof. For (a):

$$\begin{array}{ll}
 & \triangleleft\text{-ADJ} \\
\equiv & \quad \{ \text{ definition of } \triangleleft\text{-}\textsc{adj} \ \} \\
 & [\, x{\triangleright}y = y \;\Rightarrow\; x{\triangleleft}y = x \,] \\
\Leftarrow & \quad \{ \text{ taking } y \text{ for } z \ \} \\
 & [\, x{\triangleright}y = z \;\Rightarrow\; x{\triangleleft}z = x \,] \\
\equiv & \quad \{ \text{ one-point rule } \} \\
 & [\, x{\triangleleft}(x{\triangleright}y) = x \,] \\
\equiv & \quad \{ \text{ definition of } {\triangleright}{\triangleleft}\text{-}\textsc{skip} \ \} \\
 & {\triangleright}{\triangleleft}\text{-SKIP}
\end{array}$$

For (c):

$$\begin{array}{ll}
 & \triangleleft\text{-ADJ} \wedge \triangleright\text{-ADJ} \\
\equiv & \quad \{ \text{ definitions of } \triangleleft\text{-}\textsc{adj} \text{ and } \triangleright\text{-}\textsc{adj} \ \} \\
 & [\, x{\triangleright}y = y \;\Rightarrow\; x{\triangleleft}y = x \,] \;\wedge\; [\, x{\triangleleft}y = x \;\Rightarrow\; x{\triangleright}y = y \,] \\
\equiv & \quad \{ \ [\, P \wedge Q \,] \;\equiv\; [\, P \,] \wedge [\, Q \,] \ \ \} \\
 & [\, (x{\triangleright}y = y \;\Rightarrow\; x{\triangleleft}y = x) \;\wedge\; (x{\triangleleft}y = x \;\Rightarrow\; x{\triangleright}y = y) \,] \\
\equiv & \quad \{ \text{ propositional calculus } \} \\
 & [\, x{\triangleleft}y = x \;\equiv\; x{\triangleright}y = y \,] \\
\equiv & \quad \{ \text{ definition of } \textsc{adj} \ \} \\
 & \text{ADJ}
\end{array}$$

For (d):

$$\begin{array}{ll}
 & \triangleleft\text{-SKIP} \\
\equiv & \quad \{ \text{ definition of } \triangleleft\text{-}\textsc{skip} \ \} \\
 & [\, xRy \Rightarrow x{\triangleleft}y = x \,]
\end{array}$$

$\Leftarrow$ $\quad$ { weakening of $\equiv$ to 'one direction' }

$[\, x{\lhd}y = x \equiv xRy \,]$

$\equiv$ $\quad$ { definition of $\lhd$-CHAR }

$\lhd$-CHAR

For (f):

$\lhd$-CHAR $\equiv$ $\rhd$-CHAR

$\equiv$ $\quad$ { definitions of $\lhd$-CHAR and $\rhd$-CHAR }

$[\, x{\lhd}y = x \equiv xRy \,] \equiv [\, x{\rhd}y = y \equiv xRy \,]$

$\Leftarrow$ $\quad$ { $[\, P \equiv Q \,] \Rightarrow ([\, P \,] \equiv [\, Q \,])$ }

$[\, x{\lhd}y = x \equiv xRy \equiv x{\rhd}y = y \equiv xRy \,]$

$\equiv$ $\quad$ { properties of $\equiv$ }

$[\, x{\lhd}y = x \equiv x{\rhd}y = y \,]$

$\equiv$ $\quad$ { definition of ADJ }

ADJ

For (g):

$\lhd\lhd$-SKIP $\equiv$ $\lhd$-EST

$\equiv$ $\quad$ { definitions of $\lhd\lhd$-SKIP and $\lhd$-EST }

$[\, (x{\lhd}y){\lhd}y = x{\lhd}y \,] \equiv [\, (x{\lhd}y)Ry \,]$

$\Leftarrow$ $\quad$ { $[\, P \equiv Q \,] \Rightarrow ([\, P \,] \equiv [\, Q \,])$ }

$[\, (x{\lhd}y){\lhd}y = x{\lhd}y \equiv (x{\lhd}y)Ry \,]$

$\Leftarrow$ $\quad$ { taking $x{\lhd}y$ for x }

$[\, x{\lhd}y = x \equiv xRy \,]$

$\equiv$ $\quad$ { definition of $\lhd$-CHAR }

$\lhd$-CHAR

For (i):

$$
\begin{array}{ll}
 & \triangleleft\triangleright\text{-SKIP} \equiv \triangleleft\text{-EST} \\
\equiv & \quad \{ \text{ definitions of } \triangleleft\triangleright\text{-SKIP and } \triangleleft\text{-EST } \} \\
 & [\,(x\triangleleft y)\triangleright y = y\,] \equiv [\,(x\triangleleft y)Ry\,] \\
\Leftarrow & \quad \{ \ [\,P \equiv Q\,] \Rightarrow ([\,P\,] \equiv [\,Q\,]) \ \} \\
 & [\,(x\triangleleft y)\triangleright y = y \equiv (x\triangleleft y)Ry\,] \\
\Leftarrow & \quad \{ \text{ taking } x\triangleleft y \text{ for } x \ \} \\
 & [\,x\triangleright y = y \equiv xRy\,] \\
\equiv & \quad \{ \text{ definition of } \triangleright\text{-CHAR } \} \\
 & \triangleright\text{-CHAR}
\end{array}
$$

The remaining properties (b), (e), (h) and (j) are each dual to one shown above. *End of proof.*

Using this, we give a sufficient condition for maintainership that will be used in the proof of Theorem 6 below. To state this condition and that of Theorem 6, we need a few auxiliary abbreviations:

$$
\begin{array}{lcl}
\bowtie\text{-CHAR} & \equiv & (\triangleleft\text{-CHAR} \vee \triangleright\text{-CHAR}) \\
\triangleleft\bowtie\text{-SKIP} & \equiv & (\triangleleft\triangleleft\text{-SKIP} \vee \triangleleft\triangleright\text{-SKIP}) \\
\triangleright\bowtie\text{-SKIP} & \equiv & (\triangleright\triangleright\text{-SKIP} \vee \triangleright\triangleleft\text{-SKIP})
\end{array}
$$

Lemma 5: Given a constraint R and suitably typed $\triangleleft$ and $\triangleright$, if

$$\bowtie\text{-CHAR} \ \wedge \ \text{ADJ} \ \wedge \ \triangleleft\bowtie\text{-SKIP} \ \wedge \ \triangleright\bowtie\text{-SKIP}$$

then $\langle\triangleleft,\triangleright\rangle$ maintains R.

Proof. The proof will establish, from the assumptions, each of the four properties of the definition of maintainership: $\triangleleft$-EST, $\triangleright$-EST, $\triangleleft$-SKIP and $\triangleright$-SKIP. First, however, we show that the assumptions imply both $\triangleleft$-CHAR and $\triangleright$-CHAR:

$$\triangleleft\text{-CHAR} \ \wedge \ \triangleright\text{-CHAR}$$

$\equiv$ { propositional calculus }

(◁-CHAR ∨ ▷-CHAR) ∧ (◁-CHAR ≡ ▷-CHAR)

$\Leftarrow$ { Lemma 4(f) }

(◁-CHAR ∨ ▷-CHAR) ∧ ADJ

$\equiv$ { definition of ⋈-CHAR }

⋈-CHAR ∧ ADJ

For ◁-EST we now have the following proof:

◁-EST

$\Leftarrow$ { propositional calculus }

(◁◁-SKIP ≡ ◁-EST) ∧ (◁▷-SKIP ≡ ◁-EST) ∧ (◁◁-SKIP ∨ ◁▷-SKIP)

$\equiv$ { definition of ◁⋈-SKIP }

(◁◁-SKIP ≡ ◁-EST) ∧ (◁▷-SKIP ≡ ◁-EST) ∧ ◁⋈-SKIP

$\Leftarrow$ { Lemma 4(g) and (i) }

◁-CHAR ∧ ▷-CHAR ∧ ◁⋈-SKIP

By duality, also ▷-EST is a consequence of the assumptions. Finally, ◁-SKIP and ▷-SKIP are, by Lemma 4(d) and (e), immediate consequences of ◁-CHAR and ▷-CHAR. *End of proof.*

Theorem 6: Given a constraint R, for suitably typed ◁ and ▷ the following statements are all equivalent:

MNT0 : ⟨◁, ▷⟩ maintains R
MNT1 : ⋈-CHAR ∧ ◁▷-SKIP ∧ ▷◁-SKIP
MNT2 : ⋈-CHAR ∧ ADJ ∧ ◁◁-SKIP ∧ ▷▷-SKIP
MNT3 : ⋈-CHAR ∧ ◁-ADJ ∧ ◁▷-SKIP ∧ ▷▷-SKIP
MNT4 : ⋈-CHAR ∧ ▷-ADJ ∧ ▷◁-SKIP ∧ ◁◁-SKIP

Proof. By Theorem 3 the implications MNT0 $\Rightarrow$ MNTi are all immediate, so it suffices to show the implications in the reverse direction. This will be done by showing that each of the assumptions MNTi implies the condition of Lemma 5. For MNT2 this is immediate. For the others, the only assumption of Lemma 5 that is not immediately implied is ADJ. For MNT1 it is shown by:

$$\begin{array}{ll} & \text{ADJ} \\ \equiv & \quad \{ \text{ Lemma 4(c) } \} \\ & \lhd\text{-ADJ} \wedge \rhd\text{-ADJ} \\ \Leftarrow & \quad \{ \text{ Lemma 4(a) and (b) } \} \\ & \lhd\rhd\text{-SKIP} \wedge \rhd\lhd\text{-SKIP} \end{array}$$

For MNT3:

$$\begin{array}{ll} & \text{ADJ} \\ \equiv & \quad \{ \text{ Lemma 4(c) } \} \\ & \lhd\text{-ADJ} \wedge \rhd\text{-ADJ} \\ \Leftarrow & \quad \{ \text{ Lemma 4(b) } \} \\ & \lhd\text{-ADJ} \wedge \lhd\rhd\text{-SKIP} \end{array}$$

Finally, MNT4 follows by duality. *End of proof.*

Example. As an example, we show that $\langle\downarrow,\uparrow\rangle$ maintains the constraint $\leq$. First,

$$\begin{array}{ll} & \lhd\text{-CHAR} \\ \equiv & \quad \{ \text{ definition of } \lhd\text{-CHAR, instantiated for the example } \} \\ & [\, x \downarrow y = x \;\equiv\; x \leq y \,] \\ \equiv & \quad \{ \text{ property of } \downarrow \text{ } \} \\ & \mathsf{true} \end{array}$$

Next,

$$
\begin{array}{ll}
 & \triangleleft\triangleright\text{-SKIP} \\
\equiv & \quad \{ \text{ definition of } \triangleleft\triangleright\text{-SKIP, instantiated for the example } \} \\
 & [\, (x \downarrow y) \uparrow y = y \,] \\
\equiv & \quad \{ \text{ property of } \uparrow \} \\
 & [\, x \downarrow y \leq y \,] \\
\equiv & \quad \{ \text{ property of } \downarrow \} \\
 & \mathsf{true}
\end{array}
$$

By duality, also $\triangleright\triangleleft$-SKIP holds. This establishes MNT1 of Theorem 6, and thereby the maintainership. *End of example.*

3 Well-orders and selectors

In the construction of maintainers we need ways to select the "best" among various possible choices, the assumption being that some choices are better (intuitively more natural) than others, in particular according to the Principle of Least Change. It is not hard to construct cases where there are several equally good candidates, and any choice involves some degree of arbitrariness. Still, we want the selection to be as predictable as possible, at the very least deterministic (repeatable), and therefore we assume that there is some systematic and consistent way of breaking ties. In this chapter we examine some notions for modelling this, in particular well-orders and selectors. We also suggest specific tie-breaking rules by using a "small is beautiful" bias and a "leftmost first" bias.

Example. Let $\oplus$ denote addition modulo 2 on the domain $\mathbb{Z}_2 = \{0, 1\}$ ($\oplus$ is also known as "exclusive or"), and consider the constraint $\langle\!\langle\oplus\rangle\!\rangle$. If $x = \langle 0, 0\rangle$ and $y = 0$, relationship $x\langle\!\langle\oplus\rangle\!\rangle y$ holds. Now assume that y gets changed to $y' = 1$. There are two choices possible for x' such that $x'\langle\!\langle\oplus\rangle\!\rangle y'$, namely $x' = \langle 0, 1\rangle$ and $x' = \langle 1, 0\rangle$. In view of the symmetry of the situation, the choice must need be rather arbitrary. Actually, the methods described below give preference to the first choice: the leftmost field of $\langle 0, 1\rangle$ is smaller than that of $\langle 1, 0\rangle$. *End of example.*

3.1 Well-orders

The usual definition of a relation $\preceq\colon A \sim A$ being a well-order is that it is a total order, and that there exists no infinite strictly descending sequence

$$x_0 \succ x_1 \succ x_2 \succ \cdots$$

(For example, any totally ordered finite set is well-ordered.) A consequence is then that each non-empty subset s of A contains a least element with respect to $\preceq$. Here, we take the latter property as the definition, and show that the former is a consequence.

Definition (well-ordered set): A pair $\langle A, \preceq \rangle$, where A is a set and $\preceq\colon A \sim A$, is a *well-ordered* set whenever there exists a function $f\colon \mathcal{P}_{+} A \to A$ satisfying the following characterization:

$$[\, f(s) = m \;\equiv\; m \in s \wedge [\, x \in s \Rightarrow m \preceq x \,]\,]$$

Such a relation $\preceq$ is called a *well-order* on A. We will usually denote the function selecting the least element with respect to a well-order $\preceq$ by $\mathsf{min}_{\preceq}$. As usual, $x \not\preceq y$ stands for $\neg(x \preceq y)$. We write $x \prec y$ for $x \preceq y \wedge y \not\preceq x$. Furthermore, $\succeq = \breve{\preceq}$ and $\succ = \breve{\prec}$.

Lemma 7: Let $\langle A, \preceq \rangle$ be a well-ordered set. Then:

(a) $\preceq$ is reflexive
(b) $\preceq$ is transitive
(c) $\preceq$ is antisymmetric
(d) $\preceq$ is total, that is, $[\, x \preceq y \;\vee\; y \preceq x \,]$
(e) $[\, x \preceq y \;\equiv\; \mathsf{min}_{\preceq}\{x, y\} = x \,]$
(f) $[\, \mathsf{min}_{\preceq}(s) \in s \,]$
(g) $[\, x \in s \;\Rightarrow\; \mathsf{min}_{\preceq}(s) \preceq x \,]$
(h) $[\, \mathsf{min}_{\preceq}\{x\} = x \,]$
(i) each infinite sequence of elements of A is not strictly descending

Properties (a)–(d) together assert that $\preceq$ is a total order.

Proof. We prove these properties in this order: (f); (g); (h); (a); (e); (c); (b); (d).

Properties (f) and (g) follow immediately from $\mathsf{min}_{\preceq}$-characterization by instantiating $m{:=}\mathsf{min}_{\preceq}(s)$, while (h) follows from (f) by instantiating $s{:=}\{x\}$.

As for (a):

$$\begin{array}{ll}
 & [\, x \preceq x \,] \\
\equiv & \qquad \{ \text{ (h) } \} \\
 & [\, \mathsf{min}_{\preceq}\{x\} \preceq x \,] \\
\Leftarrow & \qquad \{ \text{ (g) } \} \\
 & [\, x \in \{x\} \,] \\
\equiv & \qquad \{ \text{ membership of singleton set } \} \\
 & \mathsf{true}
\end{array}$$

As for (e), which shows that $\mathsf{min}_{\preceq}$ determines $\preceq$:

$$\begin{array}{ll}
 & x \preceq y \\
\equiv & \qquad \{ \; x \in \{x,y\}, \text{ (a): } \preceq \text{ is reflexive } \} \\
 & x \in \{x,y\} \wedge x \preceq x \wedge x \preceq y \\
\equiv & \qquad \{ \text{ universal quantification over } \{x,y\} \; \} \\
 & x \in \{x,y\} \;\wedge\; [\, u \in \{x,y\} \;\Rightarrow\; x \preceq u \,] \\
\equiv & \qquad \{ \; \mathsf{min}_{\preceq}\text{-characterization } \} \\
 & \mathsf{min}_{\preceq}\{x,y\} = x
\end{array}$$

As for (c), we have to show that $[\, x = y \;\Leftarrow\; x \preceq y \wedge y \preceq x \,]$.

$$\begin{array}{ll}
 & x = y \\
\Leftarrow & \qquad \{ \; = \text{ is symmetric and transitive } \}
\end{array}$$

$$\begin{array}{ll} & \mathsf{min}_{\preceq}\{x,y\} = x \;\wedge\; \mathsf{min}_{\preceq}\{x,y\} = x \\ \equiv & \quad \{\ \text{(e)}\ \} \\ & x \preceq y \wedge y \preceq x \end{array}$$

As for (b), we have to show that $[\, x \preceq z \;\Leftarrow\; x \preceq y \wedge y \preceq z \,]$. Abbreviate $\mathsf{min}_{\preceq}\{x,y,z\}$ to m, and note that, by (g), we have $m \preceq x$, $m \preceq y$ and $m \preceq z$. Further, since $m \in \{x,y,z\}$ by (f), we have

$$m = x \;\vee\; m = y \;\vee\; m = z$$

We proceed by distinguishing these three cases.

Case $m = x$:

$$\begin{array}{ll} & x \preceq z \\ \equiv & \quad \{\ m = x,\, m \preceq z\ \} \\ & \mathsf{true} \\ \Leftarrow & \quad \{\ \text{propositional calculus}\ \} \\ & x \preceq y \wedge y \preceq z \end{array}$$

Case $m = y$:

$$\begin{array}{ll} & x \preceq z \\ \Leftarrow & \quad \{\ \text{Leibniz}\ \} \\ & x = y \wedge y \preceq z \\ \Leftarrow & \quad \{\ \text{(c): } \preceq \text{ is antisymmetric}\ \} \\ & x \preceq y \wedge y \preceq x \wedge y \preceq z \\ \equiv & \quad \{\ m = y,\, m \preceq x\ \} \\ & x \preceq y \wedge y \preceq z \end{array}$$

Case $m = z$:

$$x \preceq z$$

$\Leftarrow \quad \{ \text{ Leibniz } \}$

$x \preceq y \wedge y = z$

$\Leftarrow \quad \{ \text{ (c): } \preceq \text{ is antisymmetric } \}$

$x \preceq y \wedge y \preceq z \wedge z \preceq y$

$\equiv \quad \{ \ m = z,\ m \preceq y \ \}$

$x \preceq y \wedge y \preceq z$

As for (d), we have to show that $[\, x \preceq y \ \vee \ y \preceq x \,]$.

$[\, x \preceq y \ \vee \ y \preceq x \,]$

$\equiv \quad \{ \text{ (e) } \}$

$[\, \mathsf{min}_{\preceq}\{x, y\} = x \ \vee \ \mathsf{min}_{\preceq}\{x, y\} = y \,]$

$\equiv \quad \{ \text{ membership of set comprehension } \}$

$[\, \mathsf{min}_{\preceq}\{x, y\} \in \{x, y\} \,]$

$\equiv \quad \{ \text{ (f) } \}$

true

As for (i), let $x_0, x_1, x_2, \ldots$ be an infinite sequence of elements of A. Define $s = (x :: \exists(i :: x = x_i))$. Then

$\neg\forall(i :: x_i \succ x_{i+1})$

$\Leftarrow \quad \{ \text{ predicate calculus } \}$

$\exists(i :: x_i \preceq x_{i+1})$

$\Leftarrow \quad \{ \ x_{i+1} \in s, \text{ instantiation } \}$

$\exists(i :: [\, x \in s \Rightarrow x_i \preceq x \,])$

$\Leftarrow \quad \{ \text{ (g) } \}$

$\exists(i :: \mathsf{min}_{\preceq}(s) = x_i)$

$\equiv \quad \{ \text{ definition of } s \ \}$

$\mathsf{min}_{\preceq}(s) \in s$

$\equiv \quad \{ \text{ (f) } \}$

true

End of proof.

We now discuss how to obtain well-orders on various domains.

Tuples. Given two ordered (not necessarily well-ordered) sets $\langle A, \preceq_A\rangle$ and $\langle B, \preceq_B\rangle$, an order $\preceq$ on the product set $A \times B$ is called *product-consistent* with $\preceq_A$ and $\preceq_B$ if

$$[\, x \preceq_A x' \wedge y \preceq_B y' \;\Rightarrow\; \langle x, y\rangle \preceq \langle x', y'\rangle \,]$$

A product-consistent order is obtained by the *lexical product order* $(\preceq_A \lfloor \preceq_B)$, defined by:

$$[\, \langle x, y\rangle \; (\preceq_A \lfloor \preceq_B) \; \langle x', y'\rangle \quad\equiv\quad x \preceq_A x' \;\wedge\; (x = x' \Rightarrow y \preceq_B y') \,]$$

This can be extended to n-ary products by bracketing, as in $A \times (B \times C)$. The result does not depend on the way of bracketing. If $\preceq_A$ and $\preceq_B$ are well-orders, then so is $(\preceq_A \lfloor \preceq_B)$.

The least element according to the lexical product order is selected by:

$$\begin{array}{lll}
[\, \mathsf{min}_{(\preceq_A \lfloor \preceq_B)} x & = & \langle m, n\rangle \\
 & & \text{where} \quad m = \mathsf{min}_{\preceq_A}(\pi_L(x)) \\
 & & \qquad\quad\;\; n = \mathsf{min}_{\preceq_B}(\pi_R^{!m}(x)) \\
 & & \qquad\quad\;\; [\, b \in \pi_R^{!a}(x) \;\equiv\; \langle a, b\rangle \in x \,] \\
]
\end{array}$$

In particular,

$$[\, \mathsf{min}_{(\preceq_A \lfloor \preceq_B)}(s \times t) \;=\; (\mathsf{min}_{\preceq_A}(s), \mathsf{min}_{\preceq_B}(t)) \,]$$

Sequences. Given an ordered (not necessarily well-ordered) set $\langle A, \preceq\rangle$, the *lexical sequence order* $\preceq^*$ on the set A^* of finite sequences of A-elements is defined inductively as follows. Let x range over A and y over A^*. Denote the empty sequence by $[\,]$, and the result of prepending x to y by $x{:}y$. Then:

$$\begin{array}{l}
[\; [\,] \;\; \preceq^* \;\; y \;] \\
[\, x{:}y \;\; \not\preceq^* \;\; [\,] \;] \\
[\, x{:}y \;\; \preceq^* \; x'{:}y' \;\equiv\; \langle x, y\rangle \, (\preceq \lfloor \preceq^*) \, \langle x', y'\rangle \,]
\end{array}$$

Even if $\preceq$ is a well-order, in general the lexical sequence order thus defined is not. For take A to be the two-letter alphabet $\{\mathsf{a},\mathsf{b}\}$ with $\mathsf{a} \prec \mathsf{b}$. Then

$$\mathsf{b} \succ^* \mathsf{ab} \succ^* \mathsf{aab} \succ^* \cdots$$

However, when restricted to sequences of equal length n, the lexical order is a well-order, namely the same as the lexical order on the corresponding n-tuples. Therefore, a well-order on A^* is obtained from a well-order $\preceq$ on A by taking:

$$[\, x \preceq^{seq} y \;\equiv\; \langle \mathsf{length}(x), x\rangle \;(\leq\lfloor\preceq^*)\; \langle \mathsf{length}(y), y\rangle \,]$$

So, for the words over the alphabet $\{\mathsf{a},\mathsf{b}\}$, writing $\prec$ for $\prec^{seq}$, we get:

$$[\,] \prec \mathsf{a} \prec \mathsf{b} \prec \mathsf{aa} \prec \mathsf{ab} \prec \mathsf{ba} \prec \mathsf{bb} \prec \mathsf{aaa} \prec \cdots$$

Sets. Given a well-ordered set $\langle A, \preceq\rangle$, a well-order on the finite sets of A-elements is obtained by taking:

$$[\, x \preceq^{set} y \;\equiv\; \mathsf{sort}(x) \preceq^{seq} \mathsf{sort}(y) \,]$$

in which sort produces a sequence from its argument sorted according to $\preceq$. Writing $\prec$ for $\prec^{set}$, we get, for $A = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$ with $\mathsf{a} \preceq \mathsf{b} \preceq \mathsf{c}$:

$$\emptyset \prec \{\mathsf{a}\} \prec \{\mathsf{b}\} \prec \{\mathsf{a},\mathsf{b}\} \prec \{\mathsf{a},\mathsf{c}\} \prec \{\mathsf{b},\mathsf{c}\} \prec \{\mathsf{a},\mathsf{b},\mathsf{c}\}$$

The same method can be applied for bags (multisets).

Maps. A *B-map* over A, where A and B are sets, is a B-valued function whose domain is a subset of A. Recall that the relational embedding $\langle\!\langle f\rangle\!\rangle$ of a function f is a binary relation, and therefore a set of pairs (also called the *graph* of f). So if $\preceq_A$ and $\preceq_B$ are well-orders on A and B, we obtain a well-order on finite B-maps over A by defining:

$$[\, x \preceq^{map} y \;\equiv\; \langle\!\langle x\rangle\!\rangle \;(\preceq_A\lfloor\preceq_B)^{set}\; \langle\!\langle y\rangle\!\rangle \,]$$

For example,

$$\emptyset \prec \{\mathsf{a}\mapsto\mathsf{a}\} \prec \{\mathsf{a}\mapsto\mathsf{b}\} \prec \{\mathsf{b}\mapsto\mathsf{a}\} \prec \{\mathsf{b}\mapsto\mathsf{b}\} \prec$$
$$\{\mathsf{a}\mapsto\mathsf{a},\mathsf{b}\mapsto\mathsf{a}\} \prec \{\mathsf{a}\mapsto\mathsf{a},\mathsf{b}\mapsto\mathsf{b}\} \prec \{\mathsf{a}\mapsto\mathsf{b},\mathsf{b}\mapsto\mathsf{a}\} \prec \{\mathsf{a}\mapsto\mathsf{b},\mathsf{b}\mapsto\mathsf{b}\}$$

Numbers. The set of natural numbers $\mathbb{N}$ is well-ordered under the usual order $\leq$. The set of integers $\mathbb{Z}$ can be well-ordered by defining:

$$[\, m \preceq^{num} n \;\equiv\; \langle |m|, m\rangle \; (\leq \lfloor \geq) \; \langle |n|, n\rangle \,]$$

The use of '$\geq$' here is not a mistake, but intentional. The idea is that the least element is the smaller in absolute size if such an element exists, and otherwise the non-negative one:

$$0 \prec 1 \prec -1 \prec 2 \prec -2 \prec \cdots$$

The set of real numbers can not be well-ordered constructively, but for the purpose of this paper we assume that the set of *representable* numbers, denoted by $\mathbb{R}$ (for example, floating-point numbers) is nowhere dense, and then the same definition as for $\mathbb{Z}$ above gives a well-order on $\mathbb{R}$. (Later we shall use the fact that this also implies that for any $b \in \mathbb{R}$, if there exists any $a \in \mathbb{R}$ with $a < b$, then there is a largest value $x \in \mathbb{R}$ such that $x < b$.)

3.2 Selectors

Suppose someone claims that some function $f \colon \mathcal{P}_{+}A \to A$, when given a non-empty subset of a well-ordered set $\langle A, \preceq\rangle$, returns its least element, so the claim is that $f(s) = \mathsf{min}_{\preceq}(s)$. However, we are not given any direct information about relation $\preceq$. If the claim is correct, that relation can be reconstructed from f by using Lemma 7(e). But does the reconstruction indeed give a well-order? And does f indeed select the least element according to that well-order? We define the concept of *selector*, and show that it is necessary and sufficient that f is a selector.

Definition (selector): A function $f \colon \mathcal{P}_{+}A \to A$ is a *selector* for A whenever it satisfies the following two properties:

(a) $\quad [\, f(s) \in s \,]$

(b) $\quad [\, f(s \cup t) \;=\; f\{f(s), f(t)\} \,]$

We first derive two further properties of selectors.

Claim: If f is a selector, then

(c) $[\, f\{x\} = x \,]$
(d) $[\, s \subseteq t \wedge f(t) \in s \;\Rightarrow\; f(s) = f(t) \,]$

Proof. For (c):

$[\, f\{x\} = x \,]$

$\equiv$ { membership of singleton set }

$[\, f\{x\} \in \{x\} \,]$

$\Leftarrow$ { by instantiating $s{:=}\{x\}$ }

$[\, f(s) \in s \,]$

$\equiv$ { selector property (a) }

true

As to (d), assume that $s \subseteq t$ and $f(t) \in s$. Then:

$f(s)$

$=$ { assumption: $f(t) \in s$ }

$f(s \cup \{f(t)\})$

$=$ { selector property (b) }

$f\{f(s), f\{f(t)\}\}$

$=$ { selector property (c) }

$f\{f(s), f(t)\}$

$=$ { selector property (b) }

$f(s \cup t)$

$=$ { assumption: $s \subseteq t$ }

$f(t)$

End of proof.

Now we prove that being a selector is both necessary and sufficient for f to be equal to $\mathsf{min}_\preceq$ for some well-order $\preceq$. We use the following definition. For selector $f\colon \mathcal{P}_{\!+}A \to A$ the relation $\preceq_f\colon A \sim A$ is defined by:

$$x \preceq_f y \;\equiv\; f\{x,y\} = x$$

Theorem 8: Function $f\colon \mathcal{P}_{\!+}A \to A$ is a selector for A if and only if there exists a well-order $\preceq$ on A such that $[\,f(s) = \mathsf{min}_\preceq(s)\,]$, and then $(\preceq) = (\preceq_f)$.

Proof. (If part) For brevity we write min for $\mathsf{min}_\preceq$ below. Assume that $\langle A, \preceq\rangle$ is a well-ordered set, and that $f = \mathsf{min}$. We establish the two selector properties for f. As to (a):

$$\begin{array}{ll} & [\,f(s) \in s\,] \\ \equiv & \quad \{ \text{ assumption: } f = \mathsf{min} \text{ } \} \\ & [\,\mathsf{min}(s) \in s\,] \\ \equiv & \quad \{ \text{ Lemma 7(f) } \} \\ & \mathsf{true} \end{array}$$

For property (b):

$$\begin{array}{ll} & [\,f(s \cup t) \;=\; f\{f(s), f(t)\}\,] \\ \equiv & \quad \{ \text{ assumption: } f = \mathsf{min} \text{ } \} \\ & [\,\mathsf{min}(s \cup t) \;=\; \mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\}\,] \\ \equiv & \quad \{ \text{ } \mathsf{min}\text{-characterization } \} \\ & [\quad \mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \in s \cup t \\ & \quad\; \wedge [\,x \in s \cup t \;\Rightarrow\; \mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \preceq x\,]] \end{array}$$

We prove the two conjuncts of the last formula separately. First,

$$\begin{array}{ll} & [\,\mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \in s \cup t\,] \\ \Leftarrow & \quad \{ \text{ Lemma 7(f): } \mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \in \{\mathsf{min}(s), \mathsf{min}(t)\} \text{ } \} \\ & [\,\{\mathsf{min}(s), \mathsf{min}(t)\} \subseteq s \cup t\,] \end{array}$$

$\equiv$ { Lemma 7(f): $\mathsf{min}(s) \in s, \mathsf{min}(t) \in t$ }

true

Next,

$\mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \preceq x$

$\Leftarrow$ { Lemma 7(g): $\mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \preceq \mathsf{min}(s)$,
$\mathsf{min}\{\mathsf{min}(s), \mathsf{min}(t)\} \preceq \mathsf{min}(t)$,
Lemma 7(b): $\preceq$ is transitive }

$\mathsf{min}(s) \preceq x \vee \mathsf{min}(t) \preceq x$

$\Leftarrow$ { Lemma 7(g) }

$x \in s \vee x \in t$

$\equiv$ { membership of set union }

$x \in s \cup t$

The fact that $(\preceq) = (\preceq_f)$ is an immediate consequence of Lemma 7(e).

(Only-if part) Assume that f is a selector. To show that $\preceq_f$ is a well-order we prove that f meets the characterization

$$[\, f(s) = m \;\equiv\; m \in s \;\wedge\; [\, x \in s \;\Rightarrow\; m \preceq_f x \,]\,]$$

We prove the equivalence by mutual implication. For the $\Leftarrow$-direction:

$f(s) = m$

$\Leftarrow$ { $=$ is transitive }

$f(s) = f\{m, f(s)\} \;\wedge\; f\{m, f(s)\} = m$

$\equiv$ { definition of $\preceq_f$ }

$f(s) = f\{m, f(s)\} \;\wedge\; m \preceq_f f(s)$

$\Leftarrow$ { by instantiating $\langle s, t\rangle{:=}\langle\{m, f(s)\}, s\rangle$ in selector property (d) }

$$
\begin{array}{ll}
& \{m, f(s)\} \subseteq s \;\wedge\; f(s) \in \{m, f(s)\} \;\wedge\; m \preceq_f f(s) \\
\equiv & \quad \{ \text{ selector property (a), propositional calculus } \} \\
& m \in s \;\wedge\; (f(s) \in s \;\Rightarrow\; m \preceq_f f(s)) \\
\Leftarrow & \quad \{ \text{ by instantiating } x{:=}f(s) \text{ } \} \\
& m \in s \;\wedge\; [\, x \in s \;\Rightarrow\; m \preceq_f x \,]
\end{array}
$$

For the $\Rightarrow$-direction, assume that $f(s) = m$. Then:

$$
\begin{array}{ll}
& m \in s \;\wedge\; [\, x \in s \;\Rightarrow\; m \preceq_f x \,] \\
\equiv & \quad \{ \text{ assumption: } f(s) = m \text{ } \} \\
& f(s) \in s \;\wedge\; [\, x \in s \;\Rightarrow\; f(s) \preceq_f x \,] \\
\equiv & \quad \{ \text{ selector property (a), definition of } \preceq_f \text{ } \} \\
& [\, x \in s \;\Rightarrow\; f\{f(s), x\} = f(s) \,] \\
\equiv & \quad \{ \text{ expand left-hand side using } f(s) \in s \text{ } \} \\
& [\, \{f(s), x\} \subseteq s \wedge f(s) \in \{f(s), x\} \;\Rightarrow\; f\{f(s), x\} = f(s) \,] \\
\Leftarrow & \quad \{ \text{ by instantiating } \langle s, t\rangle{:=}\langle \{f(s), x\}, s\rangle \text{ } \} \\
& [\, s \subseteq t \wedge f(t) \in s \;\Rightarrow\; f(s) = f(t) \,] \\
\equiv & \quad \{ \text{ selector property (d) } \} \\
& \mathsf{true}
\end{array}
$$

End of proof.

The relevance of selectors is that they give a way of defining well-orders implicitly, and define precisely that which we are interested in: a function selecting an element in a systematic and consistent way.

Given sets A and B with respective selectors f_A and f_B, we call a selector f for $A \times B$ *product-consistent* with f_A and f_B whenever:

$$[\, f(s \times t) \;=\; \langle f_A(s), f_B(t)\rangle \,]$$

Claim: If f is product-consistent with f_A and f_B, then $\preceq_f$ is product-consistent with $\preceq_{f_A}$ and $\preceq_{f_B}$.

Proof. Assume that f is product-consistent with f_A and f_B. Then:

$$
\begin{array}{ll}
 & \langle x,y\rangle \preceq_f \langle x',y'\rangle \\
\equiv & \quad \{ \text{ definition of } \preceq_f \ \} \\
 & f\{\langle x,y\rangle,\langle x',y'\rangle\} \ = \ \langle x,y\rangle \\
\equiv & \quad \{ \text{ selector property (d), using} \\
 & \qquad\qquad \{\langle x,y\rangle,\langle x',y'\rangle\} \subseteq \{x,x'\}\times\{y,y'\} \\
 & \qquad\qquad \text{and } \langle x,y\rangle \in \{\langle x,y\rangle,\langle x',y'\rangle\}, \\
 & \qquad = \text{ is symmetric and transitive } \} \\
 & f(\{x,x'\}\times\{y,y'\}) \ = \ \langle x,y\rangle \\
\equiv & \quad \{ \text{ assumption: } f \text{ is product-consistent with } f_A \text{ and } f_B \ \} \\
 & \langle f_A\{x,x'\}, f_B\{y,y'\}\rangle \ = \ \langle x,y\rangle \\
\equiv & \quad \{ \text{ equality of pairs } \} \\
 & f_A\{x,x'\} \ = \ x \ \wedge \ f_B\{y,y'\} \ = \ y \\
\equiv & \quad \{ \text{ definition of } \preceq_f \ \} \\
 & x \preceq_{f_A} x' \wedge y \preceq_{f_B} y'
\end{array}
$$

End of proof.

A fact that we shall not use is that the converse holds as well: if $\preceq$ is product-consistent with $\preceq_A$ and $\preceq_B$, then $\mathsf{min}_\preceq$ is product-consistent with $\mathsf{min}_{\preceq_A}$ and $\mathsf{min}_{\preceq_B}$.

Initial element. The design rules given below for constraint maintainers assume that each linked object has a value. Therefore, when a constraint-linked network is created or extended with new objects, for constraint maintenance to be possible, each newly created object must be made to possess a value. The recommended method is to initialize it to some domain-dependent initial value, after which it should be "updated" (if it is linked by a constraint to other objects) so as to establish the constraint.

So we assume that for each non-empty set A there is a designated element of A, the *initial element* of A, denoted by $\mathsf{init}(A)$, so

(a) $\quad [\, \mathsf{init}(A) \in A \,]$

Although we do not give precise rules for determining initial elements, in general it should be the "nullest" element in some intuitively natural sense. For the domain of real numbers $\mathbb{R}$, the intuitively natural choice is $\mathsf{init}(\mathbb{R}) = 0$. For the set of all subsets $\mathcal{P}A$ of a (possibly empty) set A, the intuitively natural choice is the empty set $\emptyset$. The choice $\mathsf{init}(A) = \mathsf{min}_{\preceq}(A)$, using the relevant well-order on A as defined in Section 3.1, satisfies this criterium whenever the well-orders it is constructed from do. We get then, for example, $\mathsf{init}(\mathcal{P}_{\!+}A) = \{\mathsf{init}(A)\}$, since

$$\begin{array}{ll}
 & \mathsf{init}(\mathcal{P}_{\!+}A) \\
= & \quad \{ \ \mathsf{init} = \mathsf{min}_{\preceq} \ \} \\
 & \mathsf{min}_{\preceq}(\mathcal{P}_{\!+}A) \\
= & \quad \{ \ \#x < \#x' \ \Rightarrow \ x \prec x' \ \} \\
 & \mathsf{min}_{\preceq}(a : a \in A : \{a\}) \\
= & \quad \{ \ \{a\} \preceq \{a'\} \ \equiv \ a \preceq a' \ \} \\
 & \{\mathsf{min}_{\preceq}(A)\} \\
= & \quad \{ \ \mathsf{init} = \mathsf{min}_{\preceq} \ \} \\
 & \{\mathsf{init}(A)\}
\end{array}$$

We further assume the following two properties:

(b) $\quad [\, A \subseteq B \wedge \mathsf{init}(B) \in A \ \Rightarrow \ \mathsf{init}(A) = \mathsf{init}(B) \,]$

(c) $\quad [\, \mathsf{init}(A \times B) = \langle \mathsf{init}(A), \mathsf{init}(B) \rangle \,]$

Since $\mathbb{N} \subseteq \mathbb{R}$ and $0 \in \mathbb{N}$, it follows that $\mathsf{init}(\mathbb{N}) = 0$.

These assumptions are fulfilled if init is a "global" selector function that is product-consistent with respect to itself and itself. Again, the choice $\mathsf{init} = \mathsf{min}_{\preceq}$ satisfies these properties.

3.3 Biased selectors

We try to reduce the construction of maintainers according to the Principle of Least Change to the following problem: Approximate a given target $x \in A$ as closely as possible when the choice is constrained to a given 'choice space' (a non-empty set) $s \subseteq A$. The best approximation will be denoted as $x\S s$.

Definition (biased selector): A function $\S\colon A \times \mathcal{P}_{+}A \to A$ is a *biased selector* for A whenever it satisfies the following properties:

(a) $[\, x\S\colon \mathcal{P}_{+}A \to A \ \text{ is a selector} \,]$
(b) $[\, x \in s \ \Rightarrow\ x\S s = x \,]$

The second biased-selector property introduces the bias: if x is selectable at all, it is the preferred choice.

Biased selectors always exist — the following definition will do:

$$\begin{aligned} x\S s &= x && \text{if } x \in s,\\ &= \mathsf{init}(s) && \text{otherwise} \end{aligned}$$

Biased selectors obtained by this construction are in general not satisfactory, but the existence result only serves for use in the Corollary to the following Theorem. Below we discuss the construction of good biased selectors.

Theorem 9: Given a ditotal relation $R\colon A \sim B$, function $\lhd\colon A \times B \to A$ defined by

$$[\, x \lhd y = x\S(Ry) \,]$$

where $\S$ is some biased selector for A, is a semi-maintainer of R.

Corollary: Any ditotal relation has a maintainer.

Proof. We establish $\lhd$-EST and $\lhd$-SKIP. For $\lhd$-EST:

$$\begin{aligned} & (x \lhd y)Ry \\ \equiv \quad & \{ \text{ section } \} \end{aligned}$$

$x \lhd y \in Ry$

$\equiv$ { definition of $\lhd$ }

$x\S(Ry) \in Ry$

$\equiv$ { selector property (f), which holds by biased-selector property (a) }

true

For $\lhd$-SKIP:

$xRy \Rightarrow x \lhd y = x$

$\equiv$ { section, definition of $\lhd$ }

$x \in Ry \Rightarrow x\S(Ry) = x$

$\equiv$ { biased-selector property (b) }

true

End of proof.

As the proof shows, both biased-selector properties are needed to make this construction of a maintainer work. By performing the construction in both directions, we obtain a maintainer $\langle \lhd, \rhd \rangle$. *The idea is now that, indeed, for specific constraints maintainers will be constructed by reasoning from properties of biased selectors, using the above construction.* Thus, this theorem is of more than purely theoretical interest. However, there is no guarantee that the attempted construction will succeed; consider for example the case that the problem xRy is undecidable, of which we will see an example in Section 4.1.

Constructing biased selectors. A good method for constructing a biased selector for a set A is described in the following steps.

1. Determine an intuitively natural notion of "closeness" between pairs of elements of A, so that it is meaningful to ask whether some element x' is closer than, equally close as, or farther than, some other element x'' is to a given element x. This gives a total pre-order $\sqsubseteq_x$, that is,

a reflexive and transitive relation under which any two elements are comparable. Clearly, the only element as close to x as x itself is x, or:

$$[\, x' \sqsubseteq_x x \;\equiv\; x' = x \,]$$

If we interpret $x' \sqsubseteq_x x''$ as $\delta\langle x, x'\rangle \leq \delta\langle x, x''\rangle$, in which δ is some metric on the choice space, it must be the case that

$$[\, x \sqsubseteq_z y \wedge y \sqsubseteq_x z \;\Rightarrow\; x \sqsubseteq_y z \,]$$

In many cases the choice for the pre-order will be fairly obvious. A good metric on structured values is often the so-called edit distance. The resulting pre-order may be refined by taking account of the distance between their discrepant elements, using the lexical product order. For example, on sets,

$$[\, \{x'\} \sqsubseteq_{\{x\}} \{x''\} \;\equiv\; x' \sqsubseteq_x x'' \,]$$

2. For non-empty $s \subseteq A$, the set $C_x(s)$ of values in s that are "as close as possible" to x is formed by the minimal values under the pre-order $\sqsubseteq_x$. In a formula:

$$[\, y \in C_x(s) \;\equiv\; y \in s \wedge [\, z \in s \Rightarrow y \sqsubseteq_x z \,]\,]$$

3. If $C_x(s)$ is always non-empty, proceed to step 4. Otherwise, $\sqsubseteq_x$ is not well-founded, that is, there exist some infinite strongly descending chain

$$x' \;\sqsupset_x\; x'' \;\sqsupset_x\; x''' \;\sqsupset_x\; \cdots$$

In that case, "adjust" relation $\sqsubseteq_x$ (see below) to ensure non-emptiness.

4. Note that, by construction, $[\, C_x(s) \subseteq s \,]$. Now define

$$[\, x\S s \;=\; \mathsf{init}(C_x(s)) \,]$$

That is, if there are several equally close candidates, it is recommended to break the tie by selecting the "nullest" one.

We prove below that any function § thus constructed is a biased selector.

In discussing how to adjust problematic (ill-founded) pre-orders, we present some conditions under which no problem can occur.

Given a well-ordered set $\langle B, \preceq \rangle$ where B is non-empty, (e.g., $\langle \mathbb{N}, \leq \rangle$), we call a function $\delta\colon A \times A \to B$ a *proximity* under $\preceq$ whenever it satisfies the property

$$[\, \delta\langle x, y\rangle = \mathsf{min}_{\preceq}(B) \;\equiv\; x = y \,]$$

In particular, a metric whose codomain is (a subset of) $\mathbb{N}$ is a proximity under $\leq$.

Two proximities

$$\begin{aligned} &\delta_A\colon A \times A \to C \\ &\delta_B\colon B \times B \to D \end{aligned}$$

under respective well-orders $\preceq_C$ and $\preceq_D$ can be combined into a single proximity

$$\delta\colon (A \times B) \times (A \times B) \to C \times D$$

under any well-order that is product-consistent with $\preceq_C$ and $\preceq_D$ by defining:

$$[\, \delta\langle\langle x, y\rangle, \langle x', y'\rangle\rangle = \langle \delta_A\langle x, y\rangle, \delta_B\langle x', y'\rangle\rangle \,]$$

If $\sqsubseteq_x$ is the pre-order derived from proximity δ by

$$[\, x' \sqsubseteq_x x'' \;\equiv\; \delta\langle x, x'\rangle \preceq \delta\langle x, x''\rangle \,]$$

then $C_x(s)$ is never empty. So a possible way of adjusting a problematic $\sqsubseteq_x$ derived from a metric is to impose a well-order on the codomain of the metric.

Further, if $\sqsubseteq_x$ is a well-order, $C_x(s) = \{\mathsf{min}_{\sqsubseteq_x}(s)\}$, and is therefore also guaranteed to be non-empty. So a good way of adjusting $\sqsubseteq_x$ is to refine it (if possible) to a well-order.

Here are some examples of problematic situations, and how to resolve them.

First, assume that we try to approximate $\sqrt{2}$ as closely as possible with a rational number. An entirely reasonable pre-order is given by:

$$[\, x \sqsubseteq_{\sqrt{2}} x' \;\equiv\; |x - \sqrt{2}| \leq |x' - \sqrt{2}| \,]$$

However, the relation is not well-founded: for any x, taking $x' = \frac{u^2+2}{2}$ where $u = x \uparrow 1$, it is the case that $x \sqsupset_{\sqrt{2}} x'$. Consequently, $C_{\sqrt{2}}(s)$ is always empty. The example is not realistic in the sense that in the application of biased selectors its left argument x and the elements of its right argument s are from the same domain.

More realistic is the case that we try to approximate the real number 0 as closely as possible with a positive real number. This case can arise from a constraint $x < y$. Using the assumption that "our" real numbers are nowhere dense, but discrete approximations of the mathematical reals, the metric codomain of non-negative real numbers becomes well-ordered, and the problem disappears. However, a more principled solution is not to use such a constraint, which is only infinitesimally different from $x \leq y$, but to use either the weaker constraint $x \leq y$, or the stronger constraint $x \leq y - \varepsilon$ for some $\varepsilon > 0$ in case x is required to be truly less than y.

Finally, take the domain of words over the alphabet $\{\mathsf{a}, \mathsf{b}\}$ with $\mathsf{a} \prec \mathsf{b}$, and assume that we try to approximate the empty word $[\,]$ with a non-empty word. Using the pre-order (actually a total order)

$$[\, x \sqsubseteq_{[\,]} x' \;\equiv\; x \preceq^* x' \,]$$

is not unreasonable, but leads to empty $C_{[\,]}(s)$. Taking instead the edit distance as metric and defining $[\, x \sqsubseteq_{[\,]} x' \;\equiv\; \mathsf{length}(x) \le \mathsf{length}(x') \,]$ solves the problem.

As a last resort, it is always possible to use the discrete metric δ defined by:

$$\begin{array}{l}
[\, x = y \;\Rightarrow\; \delta\langle x, y\rangle = 0 \,] \\
[\, x \neq y \;\Rightarrow\; \delta\langle x, y\rangle = 1 \,]
\end{array}$$

This gives for C_x:

$$\begin{array}{l}
[\, x \in s \;\Rightarrow\; C_x(s) = \{x\} \,] \\
[\, x \notin s \;\Rightarrow\; C_x(s) = \;s\; \,]
\end{array}$$

We recapitulate the essential required properties of $\sqsubseteq_x$, C_x and § for some domain A, where the dummy x ranges over a, and s ranges over $\mathcal{P}_+ A$:

$$\begin{array}{l}
\sqsubseteq_x \text{ is a total pre-order} \\
[\, x' \sqsubseteq_x x \;\equiv\; x' = x \,] \\
[\, y \in C_x(s) \;\equiv\; y \in s \wedge [\, z \in s \Rightarrow y \sqsubseteq_x z \,]\,] \\
[\, x§s \;=\; \mathsf{init}(C_x(s)) \,]
\end{array}$$

Before proceeding to the main claim, we prove some consequences of these properties.

Lemma 10: For $\sqsubseteq_x$, C_x and § constructed as described above:

(a) $[\, x \sqsubseteq_x y \,]$

(b) $[\, x \in s \Rightarrow C_x(s) = \{x\} \,]$

(c) $[\, u \in s \Rightarrow x§s \sqsubseteq_x u \,]$

Proof. First we show (a):

$$
\begin{array}{ll}
 & [\, x \sqsubseteq_x y \,] \\
\equiv & \quad \{ \text{ propositional calculus: } p \;\equiv\; (p \vee q) \wedge (q \Rightarrow p) \;\} \\
 & [\, (x \sqsubseteq_x y \vee y \sqsubseteq_x x) \wedge (y \sqsubseteq_x x \Rightarrow x \sqsubseteq_x y) \,] \\
\equiv & \quad \{ \sqsubseteq_x \text{ is total } \} \\
 & [\, y \sqsubseteq_x x \Rightarrow x \sqsubseteq_x y \,] \\
\equiv & \quad \{ \text{ the only element as close to } x \text{ as } x \text{ itself is } x \;\} \\
 & [\, y = x \Rightarrow x \sqsubseteq_x y \,] \\
\equiv & \quad \{ \text{ one-point rule } \} \\
 & [\, x \sqsubseteq_x x \,] \\
\equiv & \quad \{ \sqsubseteq_x \text{ is reflexive } \} \\
 & \mathsf{true}
\end{array}
$$

For part (b), assume that $x \in s$. Then:

$$
\begin{array}{ll}
 & C_x(s) = \{x\} \\
\equiv & \quad \{ \text{ extensionality, membership of singleton set } \} \\
 & [\, y \in C_x(s) \;\equiv\; y = x \,] \\
\equiv & \quad \{ \text{ construction of } C_x(s) \;\} \\
 & [\, y \in s \;\wedge\; [\, z \in s \;\Rightarrow\; y \sqsubseteq_x z \,] \;\equiv\; y = x \,]
\end{array}
$$

We split the equivalence into an if part and an only-if part. For the if part:

$$
\begin{array}{ll}
 & [\, y = x \;\Rightarrow\; y \in s \;\wedge\; [\, z \in s \;\Rightarrow\; y \sqsubseteq_x z \,]\,] \\
\equiv & \quad \{ \text{ one-point rule } \} \\
 & x \in s \;\wedge\; [\, z \in s \;\Rightarrow\; x \sqsubseteq_x z \,] \\
\equiv & \quad \{ \text{ assumption: } x \in s \;\} \\
 & [\, z \in s \;\Rightarrow\; x \sqsubseteq_x z \,]
\end{array}
$$

$\equiv \quad \{ \text{ shown above: (a) } \}$

true

For the only-if part:

$y = x$

$\equiv \quad \{ \text{ the only element as close to } x \text{ as } x \text{ itself is } x \}$

$y \sqsubseteq_x x$

$\equiv \quad \{ \text{ assumption: } x \in s \}$

$x \in s \Rightarrow y \sqsubseteq_x x$

$\Leftarrow \quad \{ \text{ by instantiating } z{:=}x \}$

$[\, z \in s \ \Rightarrow \ y \sqsubseteq_x z \,]$

$\Leftarrow \quad \{ \text{ propositional calculus } \}$

$y \in s \ \wedge \ [\, z \in s \ \Rightarrow \ y \sqsubseteq_x z \,]$

As to part (c), assume that $u \in s$. Then:

$x \S s \sqsubseteq_x u$

$\equiv \quad \{ \text{ construction of } \S \}$

$\mathsf{init}(C_x(s)) \sqsubseteq_x u$

$\Leftarrow \quad \{ \text{ initial-element property (a): } [\, \mathsf{init}(C_x(s)) \in C_x(s) \,] \}$

$[\, y \in C_x(s) \ \Rightarrow \ y \sqsubseteq_x u \,]$

$\equiv \quad \{ \text{ construction of } C_x(s) \}$

$[\, y \in s \ \wedge \ [\, z \in s \ \Rightarrow \ y \sqsubseteq_x z \,] \ \Rightarrow \ y \sqsubseteq_x u \,]$

$\Leftarrow \quad \{ \text{ instantiate } z{:=}u \text{, assumption: } u \in s \}$

$[\, y \in s \ \wedge \ y \sqsubseteq_x u \ \Rightarrow \ y \sqsubseteq_x u \,]$

$\equiv \quad \{ \text{ propositional calculus } \}$

true

End of proof.

Now we come to the main claim of this section.

Theorem 11: Any function § constructed as described above is a biased selector.

Proof. We establish first that x§ is a selector for all x by proving the two defining selector properties. As to selector property (a),

$$
\begin{array}{ll}
 & [\, x\S s \in s \,] \\
\equiv & \qquad \{ \text{ construction of } \S \text{ } \} \\
 & [\, \mathsf{init}(C_x(s)) \in s \,] \\
\Leftarrow & \qquad \{ \ C_x(s) \subseteq s \ \} \\
 & [\, \mathsf{init}(C_x(s)) \in C_x(s) \,] \\
\equiv & \qquad \{ \text{ initial-element property (a) } \} \\
 & [\, \mathsf{true} \,]
\end{array}
$$

For selector property (b),

$$
\begin{array}{ll}
 & [\, x\S(s \cup t) \ = \ x\S\{x\S s, x\S t\} \,] \\
\equiv & \qquad \{ \text{ construction of } \S \text{ } \} \\
 & [\, \mathsf{init}(C_x(s \cup t)) \ = \ \mathsf{init}(C_x\{x\S s, x\S t\}) \,] \\
\Leftarrow & \qquad \{ \text{ initial-element property (b) } \} \\
 & [\quad C_x\{x\S s, x\S t\} \subseteq C_x(s \cup t) \ \wedge \\
 & \quad\ \ \mathsf{init}(C_x(s \cup t)) \in C_x\{x\S s, x\S t\} \\
 &]
\end{array}
$$

Again, we prove the two conjuncts separately. First,

$$
\begin{array}{ll}
 & [\, C_x\{x\S s, x\S t\} \subseteq C_x(s \cup t) \,] \\
\equiv & \qquad \{ \text{ definition of } \subseteq \text{ } \} \\
 & [\, y \in C_x\{x\S s, x\S t\} \ \Rightarrow \ y \in C_x(s \cup t) \,]
\end{array}
$$

$\equiv \quad \{$ construction of C_x $\}$

$$\begin{array}{l}[\quad y \in \{x\S s, x\S t\} \wedge [\, z \in \{x\S s, x\S t\} \Rightarrow y \sqsubseteq_x z\,] \Rightarrow \\ \quad y \in s \cup t \wedge [\, z \in s \cup t \Rightarrow y \sqsubseteq_x z\,] \\ ]\end{array}$$

To establish the validity of the implication, assume its antecedent. The conjunct $y \in \{x\S s, x\S t\}$ equivales the disjunction $y = x\S s \vee y = x\S t$. By the $s \leftrightarrow t$-symmetry, it suffices to prove the consequent under the more specific assumption that $y = x\S s \wedge [z \in \{x\S s, x\S t\} \Rightarrow y \sqsubseteq_x z]$. We first simplify this assumption:

$y = x\S s \wedge [\, z \in \{x\S s, x\S t\} \Rightarrow y \sqsubseteq_x z\,]$

$\equiv \quad \{$ membership of set union, propositional calculus $\}$

$y = x\S s \wedge [\, (z = x\S s \Rightarrow y \sqsubseteq_x z) \wedge (z = x\S t \Rightarrow y \sqsubseteq_x z)\,]$

$\equiv \quad \{$ one-point rule $\}$

$y = x\S s \wedge x\S s \sqsubseteq_x x\S s \wedge x\S s \sqsubseteq_x x\S t$

$\equiv \quad \{ \sqsubseteq_x$ is reflexive $\}$

$y = x\S s \wedge x\S s \sqsubseteq_x x\S t$

Then:

$y \in s \cup t \wedge [\, z \in s \cup t \Rightarrow y \sqsubseteq_x z\,]$

$\equiv \quad \{$ assumption: $y = x\S s$ $\}$

$x\S s \in s \cup t \wedge [\, z \in s \cup t \Rightarrow x\S s \sqsubseteq_x z\,]$

$\equiv \quad \{$ proved above: $x\S$ has selector property (a) $\}$

$[\, z \in s \cup t \Rightarrow x\S s \sqsubseteq_x z\,]$

$\equiv \quad \{$ membership of set union, propositional calculus, $[\, P \wedge Q\,] = [\, P\,] \wedge [\, Q\,]$ $\}$

$[\, z \in s \Rightarrow x\S s \sqsubseteq_x z\,] \wedge [\, z \in t \Rightarrow x\S s \sqsubseteq_x z\,]$

$\Leftarrow \quad \{$ assumption: $x\S s \sqsubseteq_x x\S t$, $\sqsubseteq_x$ is transitive $\}$

$[\, z \in s \Rightarrow x\S s \sqsubseteq_x z \,] \ \wedge \ [\, z \in t \Rightarrow x\S t \sqsubseteq_x z \,]$

$\equiv$ { Lemma 10(c) }

true

We have proved now that $x\S$ is a selector, thereby establishing biased-selector property (a) for §.

As to biased-selector property (b):

$x\S s = x$

$\equiv$ { construction of § }

$\mathsf{init}(C_x(s)) = x$

$\Leftarrow$ { membership of singleton set }

$\mathsf{init}(C_x(s)) \in \{x\}$

$\Leftarrow$ { initial-element property (a): $\mathsf{init}(C_x(s)) \in C_x(s)$ }

$C_x(s) = \{x\}$

$\Leftarrow$ { Lemma 10(b) }

$x \in s$

End of proof.

4 Combining maintainers

Assume that we have maintainers of constraints R and S. The question we consider here is whether these maintainers can somehow be combined to form a maintainer of some composition $R \oplus S$ of these constraints.

4.1 Relational composition

Given maintainers of constraints R: $A \sim B$ and S: $B \sim C$, can they be combined to form a maintainer of their composition $(R\,;S)$: $A \sim C$?

We first show that in the general case this is impossible, at least in a constructive way. The assumption of constructiveness implies that, if the constituent maintainers are computable, so is the combined maintainer. Further, if a relation has a computable maintainer, or even only a compuable semi-maintainer, the relation is decidable by virtue of $\triangleleft$-EST or dually $\triangleright$-EST. Below we present two relations that both have computable maintainers, but whose composition is undecidable. So no maintainer of the composition can be constructed from maintainers of the constituents.

Let $\mathcal{CFG}$ be the set of context-free grammars over some alphabet, with the exclusion of grammars whose language is empty. Let $\mathcal{S}$ be the set of words over the same alphabet. Consider constraint $\vdash$: $\mathcal{CFG} \sim \mathcal{S}$, where $G \vdash x$ means: grammar G has word x as a terminal production, or, equivalently, $x \in \mathcal{L}(G)$, the language produced by G. It is easy to find a maintainer of $\vdash$; take, e.g.,

$$\begin{array}{rcll} G\triangleleft x & = & G & \text{if } G \vdash x, \\ & = & \{\mathsf{S} \to x\} & \text{otherwise} \end{array}$$

$$\begin{array}{rcll} G\triangleright x & = & x & \text{if } G \vdash x, \\ & = & \mathsf{init}(\mathcal{L}(G)) & \text{otherwise} \end{array}$$

Here, $\{\mathsf{S} \to x\}$ stands for a context-free grammar whose only production from the start symbol S gives word x. So $\vdash$, and by duality $\breve{\vdash}$, have computable maintainers. But the composition $(\vdash;\breve{\vdash})$ is an undecidable relation, since

$$G0 (\vdash;\breve{\vdash})\, G1$$

$$\equiv \quad \{ \text{ relational composition } \}$$

$$\exists (x : x \in S : G0 \vdash x \wedge x \breve{\vdash} G1)$$

$$\equiv \quad \{ \text{ converse } \}$$

$$\begin{array}{ll}
& \exists(x : x \in S : \mathit{G0}\vdash x \wedge \mathit{G1}\vdash x) \\
\equiv & \quad \{ \text{ equivalently } \} \\
& \exists(x : x \in S : x \in \mathcal{L}(\mathit{G0}) \wedge x \in \mathcal{L}(\mathit{G1})) \\
\equiv & \quad \{ \text{ membership of intersection } \} \\
& \exists(x : x \in S : x \in \mathcal{L}(\mathit{G0}) \cap \mathcal{L}(\mathit{G1})) \\
\equiv & \quad \{ \text{ membership of empty set } \} \\
& \mathcal{L}(\mathit{G0}) \cap \mathcal{L}(\mathit{G1}) \neq \emptyset
\end{array}$$

The latter proposition, common membership of two context-free languages, is a well-known undecidable problem.

In the context of using maintainers to guard the consistency of a constrained tree, this impossibility result is not a great loss. In order to create an $(R\,;S)$ constraint between two objects A and C, it suffices to introduce a mediating object B and to create an R constraint between A and B, and an S constraint between B and C.

The following theorem deals with an interesting special case in which maintainers of a composition $(R\,;S)$ *can* be created from maintainers of the individual components, namely when R (or, dually, $\breve{S}$) is functional.

Theorem 12: Given a maintainer $\langle \lhd_F , \rhd_F \rangle$ of a functional relation $F = \langle\!\langle f \rangle\!\rangle\colon A \sim B$, and a maintainer $\langle \lhd_S , \rhd_S \rangle$ of an arbitrary constraint $S\colon B \sim C$, where $B \neq \emptyset$, the composition $(F\,;S)$ of these constraints is maintained by $\langle \lhd, \rhd \rangle$, where

$$\begin{array}{l}
[\, x \lhd z \;=\; x \lhd_F (f(x) \lhd_S z) \,] \\
[\, x \rhd z \;=\; f(x) \rhd_S z \,]
\end{array}$$

(If f is not already explicitly given, it can be retrieved, using Theorem 1, by $[\, f(x) = (x \rhd_F (\mathsf{init}(B))) \,]$.)

Proof. We establish MNT1, and thereby, by Theorem 6, the maintainership. To establish MNT1, we establish $\rhd$-CHAR, $\lhd\rhd$-SKIP and $\rhd\lhd$-SKIP. First, however, we derive the auxiliary result $[\, f(x \lhd z) = f(x) \lhd_S z \,]$, which is used in the proof of $\lhd\rhd$-SKIP:

$$[\, f(x \lhd_F y) = y \,]$$

$\equiv$ $\{$ definition of F $\}$

$[\,(x \lhd_F y) F y\,]$

$\equiv$ $\{$ $\lhd$-EST for $\lhd_F$ $\}$

true

and so

$f(x \lhd z)$

$=$ $\{$ definition of $\lhd$ $\}$

$f(x \lhd_F (f(x) \lhd_S z))$

$=$ $\{$ result just above with $y := f(x) \lhd_S z$ $\}$

$f(x) \lhd_S z$

For $\rhd$-CHAR we argue:

$x \rhd z = z$

$\equiv$ $\{$ definition of $\rhd$ $\}$

$f(x) \rhd_S z = z$

$\equiv$ $\{$ $\rhd$-CHAR for $\rhd_S$ $\}$

$f(x) S z$

$\equiv$ $\{$ one-point rule for $\exists$ $\}$

$\exists(y :: f(x) = y \wedge y S z)$

$\equiv$ $\{$ definition of F $\}$

$\exists(y :: x F y \wedge y S z)$

$\equiv$ $\{$ definition of relational composition $\}$

$x(F\,;S)z$

For $\lhd\rhd$-SKIP:

$(x \lhd z) \rhd z$

$=$ $\{$ definition of $\triangleright$ $\}$

$f(x \triangleleft z) \triangleright_S z$

$=$ $\{$ auxiliary result $\}$

$(f(x) \triangleleft_S z) \triangleright_S z$

$=$ $\{$ $\triangleleft\triangleright$-SKIP for $\langle \triangleleft_S , \triangleright_S \rangle$ $\}$

z

Finally, for $\triangleright\triangleleft$-SKIP:

$x \triangleleft (x \triangleright z) = x$

$\equiv$ $\{$ definitions of $\triangleleft$ and $\triangleright$ $\}$

$x \triangleleft_F (f(x) \triangleleft_S (f(x) \triangleright_S z)) = x$

$\equiv$ $\{$ $\triangleright\triangleleft$-SKIP for $\langle \triangleleft_S , \triangleright_S \rangle$ $\}$

$x \triangleleft_F f(x) = x$

$\equiv$ $\{$ $\triangleleft$-CHAR for $\triangleleft_F$ $\}$

$x F f(x)$

$\equiv$ $\{$ definition of F $\}$

true

End of proof.

These maintainers require, each time, the recomputation of $f(x)$. If that involves an expensive computation, it is more efficient to implement $(F\,;S)$ with a mediating object that caches the function results. But if f is cheap to compute, the implementation suggested by this theorem saves some administrative overhead.

In a case like $(F\,;\breve{G})$, where $F = \langle\!\langle f \rangle\!\rangle$ and $G = \langle\!\langle g \rangle\!\rangle$, Theorem 12 gives us two ways of computing a maintainer. The result, however, is the same for either way. We show this by following one way; the result is obviously dual to the effect of swapping F and G:

$$x \lhd z$$

$= \quad \{$ Theorem 12 $\}$

$$x \lhd_F (f(x) \lhd_{\breve{G}} z)$$

$= \quad \{$ duality $\}$

$$x \lhd_F (z \lhd_G f(x))$$

$= \quad \{$ Theorem 1 $\}$

$$x \lhd_F g(z)$$

while

$$x \rhd z$$

$= \quad \{$ Theorem 12 $\}$

$$f(x) \rhd_{\breve{G}} z$$

$= \quad \{$ duality $\}$

$$z \rhd_G f(x)$$

Decomposing constraints. Sometimes finding a maintainer of some constraint R is problematic, while it is possible to find maintainers of S and T, where $R = (S\,;T)$. While we have just seen that this does not help us to construct a *maintainer* of R (unless S or $\breve{T}$ is functional), it gives an *implementation* for the constraint R between two objects by means of a mediating object, as sketched in Figure 3.

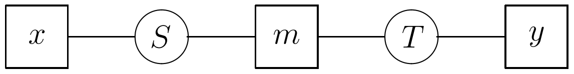

Figure 3: The constraint $x(S\,;T)y$ implemented by the two constraints xSm and mTy, using a mediating object m.

An example of this phenomenon is, of course, given by $R = (\vdash ; \breve{\vdash})$ treated above. The result of this construction does not necessarily follow the Principle of Least Change, even when the constituent maintainers do. For example, take the top relation $\mathbb{T}$: $\mathbb{Z} \sim \mathbb{Z}$, of which the generic version is treated in 5.1.b. Since the constraint is always satisfied, a change to one side should entail no change to the other side. It can be decomposed by $\mathbb{T} = (\neq ; \neq)$, where $\neq$: $\mathbb{Z} \sim \mathbb{Z}$ is treated in 5.4.a. Suppose now that constraint $\mathbb{T}$ between x and y is implemented according to this decomposition with a mediating object m, and that the initial values are $x = 0, m = 1, y = 0$. If now x gets changed to 1, the maintainer of the first inequality constraint will change m to 0. Now the second inequality constraint is violated, and y gets set to 1.

In spite of the potential violation of the Principle of Least Change, using a mediating object is (sometimes) a useful construction, in particular when direct methods fail. Furthermore, as we shall see later, such decompositions may reflect the structure of a constraint in a natural way.

We treat one standard decomposition, other useful decompositions usually being expressible as specialized variants of the standard one. Let R: $A \sim B$. In the typings given below we view R as a set of pairs; more specifically, a subset of $A \times B$. Define the constraints

$$S\colon A \sim R$$
$$T\colon R \sim B$$

by:

$$[\, xS\langle \xi, \eta \rangle \;\equiv\; x = \xi \wedge \xi R \eta \,]$$
$$[\, \langle \xi, \eta \rangle T y \;\equiv\; \xi R \eta \wedge \eta = y \,]$$

In both cases the typing already implies that $\langle \xi, \eta \rangle \in R$, so that we could remove the conjunct $\xi R \eta$ in either definiens above without change of meaning. The reason it is explicitly included, though, is that this allows to fold instances of the definiens without having to check the typing prerequisites, while in the other direction it gives a verification of the typing of proposed maintainers. First we show that indeed $R = (S ; T)$:

$$xRy$$

$\equiv \quad \{$ one-point rules for $\exists$ $\}$

$\exists(\xi, \eta :: x = \xi \wedge \xi R \eta \wedge \eta = y)$

$\equiv \quad \{$ predicate calculus $\}$

$\exists(\xi, \eta :: (x = \xi \wedge \xi R \eta) \wedge (\xi R \eta \wedge \eta = y))$

$\equiv \quad \{$ definitions of S and T $\}$

$\exists(\xi, \eta :: xS\langle \xi, \eta \rangle \wedge \langle \xi, \eta \rangle T y)$

$\equiv \quad \{$ relational composition $\}$

$x(S\,;T)y$

The constraints $\breve{S}$ and T are both functional, and by Theorem 1 we have no choice but to define

$$[\, x \triangleleft_S \langle \xi, \eta \rangle \;=\; \xi \,]$$
$$[\, \langle \xi, \eta \rangle \triangleright_T y \;=\; \eta \,]$$

Suppose that we only have a *semi*-maintainer of R. This then gives us a bidirectional maintainer of S or T, depending on the direction. We treat one case, the other being dual.

Theorem 13: Let R and S be as above, and let $\triangleright_R$ be a semi-maintainer of R (i.e., satisfying $\triangleright$-EST and $\triangleright$-SKIP). Then $\langle \triangleleft_S, \triangleright_S \rangle$ maintains S, where $\triangleleft_S$ is as above, and $\triangleright_S$ is given by:

$$[\, x \triangleright_S \langle \xi, \eta \rangle \;=\; \langle x, x \triangleright_R \eta \rangle \,]$$

Proof. We only have to verify $\triangleright$-EST and $\triangleright$-SKIP for $\triangleright_S$.

For $\triangleright$-EST:

$xS(x \triangleright_S \langle \xi, \eta \rangle)$

$\equiv \quad \{$ definition of $\triangleright_S$ $\}$

$xS\langle x, x \triangleright_R \eta \rangle$

$\equiv \quad \{$ definition of S $\}$

$$x = x \wedge xR(x \rhd_R \eta)$$

$\equiv$ { $=$ is reflexive, $\rhd$-EST for $\rhd_R$ }

$$\mathsf{true}$$

For $\rhd$-SKIP:

$$x \rhd_S \langle \xi, \eta \rangle = \langle \xi, \eta \rangle$$

$\equiv$ { definition of $\rhd_S$ }

$$\langle x, x \rhd_R \eta \rangle = \langle \xi, \eta \rangle$$

$\equiv$ { pairing is surjective }

$$x = \xi \wedge x \rhd_R \eta = \eta$$

$\equiv$ { $\rhd$-CHAR for $\rhd_R$ }

$$x = \xi \wedge xR\eta$$

$\equiv$ { one-point rule }

$$x = \xi \wedge \xi R\eta$$

$\equiv$ { definition of S }

$$xS\langle \xi, \eta \rangle$$

End of proof.

4.2 Relational union

Given maintainers of constraints R: $A \sim B$ and S: $A \sim B$, can they be combined to form a maintainer of their union $R \cup S$: $A \sim B$?

We use the fact that $\cup$ distributes through sectioning, since:

$$(R \cup S)y$$

$=$ { section }

$$(x :: x(R \cup S)y)$$

$=$ { switching notation }

$(x :: \langle x, y\rangle \in R{\cup}S)$

$=$ { definition of $\cup$ }

$(x :: \langle x, y\rangle \in R \vee \langle x, y\rangle \in S)$

$=$ { switching notation }

$(x :: xRy \vee xSy)$

$=$ { definition of $\cup$ }

$(x :: xRy) \cup (x :: xSy)$

$=$ { section }

$(Ry) \cup (Sy)$

Because of the duality of $\cup$ — that is, $(R{\cup}S)\breve{} = \breve{R}{\cup}\breve{S}$ — we focus on finding a semi-maintainer $\lhd$. We can indeed find a maintainer under the following assumption: *Both $\lhd_R$ and $\lhd_S$ are based on the* same *biased selector* §*, using the construction of Theorem 9.* For then, continuing that construction,

$x{\lhd}y$

$=$ { Theorem 9 }

$x\S((R{\cup}S)y)$

$=$ { $\cup$ distributes through sectioning }

$x\S((Ry) \cup (Sy))$

$=$ { selector property (b) }

$x\S\{x\S(Ry), x\S(Sy)\}$

$=$ { construction of $\lhd_R$ and $\lhd_S$ }

$x\S\{x{\lhd_R}y, x{\lhd_S}y\}$

Example. We consider relations of type $\mathbb{Z} \sim \mathbb{Z}$. Let $[\, xRy \equiv x \le y-1 \,]$ and $[\, xSy \equiv x \ge y+1 \,]$, with semi-maintainers

$$\begin{array}{rcl} x \lhd_R y & = & x \downarrow (y-1) \\ x \lhd_S y & = & x \downarrow (y+1) \end{array}$$

Because we are in the domain of integers, $R \cup S = (\neq)$. In the construction of $\lhd$ as above we distinguish two cases.

The first case is $x \neq y$. In this case, either $x \le y-1$ or $x \ge y+1$, so either $x \lhd_R y = x$ or $x \lhd_S y = x$. Then $x \lhd y = x$ by biased-selector property (b). (The same result could, of course, have immediately been obtained from $\lhd$-SKIP.)

The second case is $x = y$. Then $x \lhd_R y = x-1$ and $x \lhd_S y = x+1$. So $x \lhd y = x\S\{x-1, x+1\}$. Both possible choices are equally close to x. Using the preferred method for tie breaking, we have $x\S\{x-1, x+1\} = \mathsf{min}\{x-1, x+1\}$, where the minimum is taken under the well-order $\preceq^{num}$, giving $x-1$ when $x > 0$, and $x+1$ otherwise. *End of example.*

5 Maintainers for specific constraints

In this section we construct maintainers of specific constraints. We give, for each constraint, first the relation, next its maintainer, and finally a justification of the construction. We reason from properties of biased selectors constructed in accordance with the methods of the last section, by using, for a ditotal relation R: $A \sim B$:

$$\begin{array}{l} [\, x \lhd y = x\S_A(Ry) \,] \\ [\, x \rhd y = y\S_B(xR) \,] \end{array}$$

For a functional relation $\langle\!\langle f \rangle\!\rangle$ we use of course Theorem 1:

$$[\, x \rhd y = f(x) \,]$$

Whenever applicable, we will assume that selector $\langle x, y\rangle\S_{A\times B}$ is product-consistent with $x\S_A$ and $y\S_B$, that is:

$$[\, \langle x, y\rangle\S_{A\times B}(s \times t) \ = \langle x\S_A s, y\S_B t\rangle \,]$$

5.1 Generic constraints

5.1.a Equality

Relation: $= : A \sim A$

Maintainer: $[\, x \triangleleft y = y \,]$
$[\, x \triangleright y = x \,]$

Equality is as tight a constraint as possible. The relation is functional: it equals $\langle\!\langle \mathsf{id} \rangle\!\rangle$, in which id is the identity function; moreover, it is symmetric. Theorem 1 now gives us the result.

5.1.b Top

Relation: $\mathbb{T} : A \sim B$

Maintainer: $[\, x \triangleleft y = x \,]$
$[\, x \triangleright y = y \,]$

Relation $\mathbb{T}$ is the top of the lattice of relations, defined by $[\, x \mathbb{T} y \,]$: anything goes. This is the laxest constraint possible; in everyday language it would not be called a constraint, and x and y would be called "unrelated". We have $[\, \mathbb{T} y \;=\; A \,]$ and $[\, x \mathbb{T} \;=\; B \,]$. Since $x \in A$ and $y \in B$, the result follows by biased-selector property (b).

It is interesting to compare this maintainer to that for equality; it is in some sense its opposite (as is the constraint). The constraint is included in this list mainly for its theoretical interest.

5.1.c Left projection

Relation: $\langle\!\langle \pi_L \rangle\!\rangle : A \times B \sim A$

Maintainer: $[\, \langle x, y \rangle \triangleleft u \;=\; \langle u, y \rangle \,]$
$[\, \langle x, y \rangle \triangleright u \;=\; x \,]$

We have

$$\begin{array}{ll}
& \langle\!\langle \pi_L \rangle\!\rangle u \\
= & \quad \{ \text{ section, } \langle\!\langle f \rangle\!\rangle \text{ and } \pi_L \ \} \\
& (x, y : \langle x, y\rangle \in A \times B : x = u) \\
= & \quad \{ \text{ membership of product set, shunting } \} \\
& (x, y :: x = u \wedge x \in A \wedge y \in B) \\
= & \quad \{ \text{ one-point rules, } u \in A \ \} \\
& (x, y :: x \in \{u\} \wedge y \in B) \\
= & \quad \{ \text{ membership of product set, set comprehension } \} \\
& \{u\} \times B
\end{array}$$

Then

$$\begin{array}{ll}
& \langle x, y\rangle \triangleleft u \\
= & \quad \{ \text{ construction of } \S \ \} \\
& \langle x, y\rangle \S (\langle\!\langle \pi_L \rangle\!\rangle u) \\
= & \quad \{ \text{ above result } \} \\
& \langle x, y\rangle \S (\{u\} \times B) \\
= & \quad \{ \text{ product-consistency } \} \\
& \langle x \S \{u\}, y \S B\rangle \\
= & \quad \{ \ y \in B, \text{ biased-selector properties } \} \\
& \langle u, y\rangle
\end{array}$$

Further, by Theorem 1, $\langle x, y\rangle \triangleright u = \pi_L \langle x, y\rangle = x$.

5.1.d Right projection

Relation: $\langle\!\langle \pi_R \rangle\!\rangle \colon A \times B \sim B$

Maintainer: $[\ \langle x, y\rangle \triangleleft v \ = \ \langle x, v\rangle \ ]$
$[\ \langle x, y\rangle \triangleright v \ = \ y \ ]$

This is the mirror image of π_L.

5.2 Constraints involving sets

For measuring the proximity of two finite sets s and t we use the size $\#(s\Delta t)$ of their symmetric set difference, where the symmetric difference of two sets is given by:

$$[\, s\Delta t \;=\; (x :: x \in s \;\;\not\equiv\;\; x \in t)\,]$$

Pleasant properties of Δ are that it is symmetric (hence the name) and associative, has $\emptyset$ as neutral element, and is involutive (self-cancelling):

$$[\, s\Delta t = t\Delta s\,]$$
$$[\, s\Delta(t\Delta u) = (s\Delta t)\Delta u\,]$$
$$[\, s\Delta\emptyset = s = \emptyset\Delta s\,]$$
$$[\, s\Delta s = \emptyset\,]$$

It follows that

$$[\, s\Delta t = u \;\equiv\; s = u\Delta t\,]$$

Further, $\cap$ distributes over Δ:

$$[\, s \cap (t\Delta u) = (s{\cap}u)\;\Delta\;(t{\cap}u)\,]$$

The normal, asymmetric set difference operation $s - t$ can be expressed as $(s{\cup}t)\,\Delta\,t$, since

$$\begin{array}{ll}
 & s - t \;=\; (s{\cup}t)\,\Delta\,t \\
\equiv & \quad \{\text{ set extensionality }\} \\
 & [\, x \in s - t \;\equiv\; x \in (s{\cup}t)\,\Delta\,t\,] \\
\equiv & \quad \{\text{ membership rules }\} \\
 & [\, (x{\in}s \wedge x{\notin}t) \;\equiv\; ((x{\in}s \vee x{\in}t) \;\not\equiv\; x{\in}t)\,] \\
\equiv & \quad \{\text{ propositional calculus }\} \\
 & \mathsf{true}
\end{array}$$

Alternative expressions for $s - t$ in terms of Δ are $s \,\Delta\, (s{\cap}t)$ and $s \cap (s\Delta t)$. Since for set $s \in \mathcal{P}A$ its set complement $\overline{s}$ with respect to A equals $A - s$, we can also express this as $A\Delta s$. These and further set-calculus properties are introduced without proof; they can be readily verified by translating them into their propositional counterparts by set extensionality, and verifying those using propositional calculus, as was done above for $s - t = (s{\cup}t) \,\Delta\, t$.

Some inequations involving Δ can be solved for unknown u thus:

$$\begin{array}{lcl} [\, s \subseteq t\Delta u & \equiv & s - t \;\subseteq\; u \;\subseteq\; \overline{s \cap t}\,] \\ [\, s\Delta u \subseteq t & \equiv & s - t \;\subseteq\; u \;\subseteq\; s \cup t\,] \end{array}$$

5.2.a Member

Relation: $\in : A \sim \mathcal{P}_{\!+}A$

Maintainer: $[\, x{\lhd}s \;=\; x\S_A s\,]$
$[\, x{\rhd}s \;=\; s \cup \{x\}\,]$

Assuming the availability of a biased selector for A, finding a definition for $\lhd$ is easy: since

$$\begin{array}{ll} & \in s \\ = & \quad \{\text{ section }\} \\ & (x :: x \in s) \\ = & \quad \{\text{ set comprehension }\} \\ & s \end{array}$$

we have

$$\begin{array}{ll} & x{\lhd}s \\ = & \quad \{\text{ construction of } \S \text{ }\} \\ & x\S_A(\in s) \\ = & \quad \{\text{ just shown: } \in s \;=\; s \text{ }\} \end{array}$$

$$x \S_A s$$

For the other direction, put $s' = x \triangleright s$. The hard requirement on s' is that $x \in s'$, or, equivalently, $\{x\} \subseteq s'$. We translate this into a requirement on the difference $d = s \Delta s'$, which we want to keep as small as possible:

$$\begin{array}{ll}
 & \{x\} \subseteq s' \\
\equiv & \quad \{ \text{ properties of } \Delta \ \} \\
 & \{x\} \subseteq s \Delta s \Delta s' \\
\equiv & \quad \{ \text{ definition of } d \ \} \\
 & \{x\} \subseteq s \Delta d \\
\equiv & \quad \{ \text{ solving for } d \ \} \\
 & \{x\} - s \subseteq d \subseteq \overline{\{x\} \cap s}
\end{array}$$

The smallest d satisfying this requirement is $d = \{x\} - s$. Then

$$\begin{array}{ll}
 & x \triangleright s \\
= & \quad \{ \text{ definition of } s' \ \} \\
 & s' \\
= & \quad \{ \text{ properties of } \Delta\text{, definition of } d \ \} \\
 & s \Delta d \\
= & \quad \{ \text{ above formula for } d \ \} \\
 & s \Delta (\{x\} - s) \\
= & \quad \{ - \text{ in terms of } \Delta \ \} \\
 & s \Delta s \Delta (s \cup \{x\}) \\
= & \quad \{ \ \Delta \text{ is involutive } \} \\
 & s \cup \{x\}
\end{array}$$

5.2.b Subset

Relation: $\subseteq : \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\, s \lhd t \;=\; s \cap t \,]$
$[\, s \rhd t \;=\; s \cup t \,]$

Putting $s' = s \lhd t$, the requirement is $s' \subseteq t$. Using $d = s \Delta s'$ we have:

$$\begin{array}{ll} & s' \subseteq t \\ \equiv & \quad \{ \text{ properties of } \Delta, \text{ definition of } d \} \\ & s \Delta d \subseteq t \\ \equiv & \quad \{ \text{ solving for } d \} \\ & s - t \;\subseteq\; d \;\subseteq\; s \cup t \end{array}$$

The smallest d satisfying this requirement is $d = s - t$. Then

$$\begin{array}{ll} & s \lhd t \\ = & \quad \{ \text{ definition of } s', \text{ properties of } \Delta, \text{ definition of } d \} \\ & s \Delta d \\ = & \quad \{ \text{ above formula for } d \} \\ & s \Delta (s - t) \\ = & \quad \{ - \text{ in terms of } \Delta \} \\ & s \,\Delta\, s \,\Delta\, (s \cap t) \\ = & \quad \{ \Delta \text{ is involutive } \} \\ & s \cap t \end{array}$$

For the other direction we find, likewise, that $s - t$ is the smallest possible value for $t \Delta t'$, giving $t' = s \cup t$.

5.2.c Disjoint

Relation: $\nmid : \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\, s \triangleleft t \;=\; s - t \,]$
$[\, s \triangleright t \;=\; t - s \,]$

This is the constraint of two sets being disjoint: $[\, s \nmid t \;\equiv\; s \cap t = \emptyset \,]$. Using the law $[\, s \cap t = \emptyset \;\equiv\; s \subseteq \overline{t} \,]$ in combination with Theorem 12, we find that $s \triangleleft_{\nmid} t \;=\; s \triangleleft_{\subseteq} \overline{t} \;=\; s \cap \overline{t} \;=\; s - t$. The symmetry of $\nmid$ gives the result for the other direction.

5.2.d Intersection

Relation: $\langle\!\langle \cap \rangle\!\rangle \colon \mathcal{P}A \times \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\, \langle s, t \rangle \triangleleft u \;=\; \langle u \cup (s{-}t), u \cup t \rangle \,]$
$[\, \langle s, t \rangle \triangleright u \;=\; s \cap t \,]$

Note the asymmetry in the definition for $\triangleleft$, in spite of the symmetry of $\cap$. It will emerge from the calculations as a result of tie breaking, which in this case also breaks the symmetry. We must tackle the construction of $\langle s', t' \rangle$ "close" to $\langle s, t \rangle$ such that $s' \cap t' = u$. For the edit distance between $\langle s, t \rangle$ and $\langle s', t' \rangle$ we use $\#d_s + \#d_t$, where $\langle d_s, d_t \rangle = \langle s \Delta s', t \Delta t' \rangle$. We know that s' and t' have to satisfy $u \subseteq s'$ and $u \subseteq t'$, since

$$
\begin{array}{ll}
 & u \subseteq s' \wedge u \subseteq t' \\
\equiv & \quad \{ \text{ set calculus } \} \\
 & u \subseteq s' \cap t' \\
\Leftarrow & \quad \{ \subseteq \text{ is reflexive } \} \\
 & s' \cap t' = u
\end{array}
$$

As before, this implies that $d_s \supseteq u - s$ and $d_t \supseteq u - t$, and so $d_s \cap d_t \supseteq (u{-}s) \cap (u{-}t) \;=\; u - (s{\cup}t)$, and $d_s \cup d_t \;\supseteq\; (u{-}s) \cup (u{-}t) \;=\; u - (s{\cap}t)$. Furthermore,

$$
\begin{array}{ll}
 & d_s \cup d_t \\
= & \quad \{ \text{ definitions of } d_s \text{ and } d_t \text{ } \}
\end{array}
$$

$(s\Delta s') \cup (t\Delta t')$

$\supseteq \quad \{$ set calculus $\}$

$(s{\cap}t) - (s'{\cap}t')$

$= \quad \{\ s' \cap t' = u\ \}$

$(s{\cap}t) - u$

Combining this with the previous lower bound on $d_s \cup d_t$, we have $d_s \cup d_t \supseteq (u - (s{\cap}t)) \cup ((s{\cap}t) - u) = u \,\Delta\, (s{\cap}t)$ (see Figure 4), and so $\#d_s + \#d_t = \#(d_s \cup d_t) + \#(d_s \cap d_t) \geq \#(u \,\Delta\, (s{\cap}t)) + \#(u - (s{\cup}t))$. If we can choose d_s

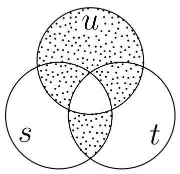

Figure 4: A Venn diagram indicating by shading the area corresponding to the lower bound $u\Delta(s{\cap}t)$ on $d_s \cup d_t$. The upper shaded part corresponds to $u - (s{\cap}t)$, the shield-shaped lower shaded part to $(s{\cap}t) - u$.

and d_t so that $d_s \cup d_t = u \,\Delta\, (s{\cap}t)$ and $d_s \cap d_t = u - (s{\cup}t)$, we get equalities for $\geq$, so that is the best choice we can hope for. Let the "jointly required" set $(s{\cap}t) - u$, that is, the set that the union $d_s \cup d_t$ is required to contain beyond the union $u - (s{\cap}t)$ of the individual lower bounds on d_s and d_t (the shield-shaped part in Figure 4), be split into two disjoint (possibly empty) sets x_s and x_t. So assume that $x_s \cup x_t = (s{\cap}t) - u$, where $x_s \nparallel x_t$. Now consider the choice $d_s = (u{-}s) \cup x_s, d_t = (u{-}t) \cup x_t$. If allowed — that is, if it satifies the requirement $s' \cap t' = u$ — this choice is optimal, since then

$d_s \cup d_t$

$= \quad \{$ choice of d_s, d_t $\}$

$(u{-}s) \cup x_s \cup (u{-}t) \cup x_t$

$= \quad \{$ set calculus $\}$

$(u - (s{\cap}t)) \cup x_s \cup x_t$

$$
\begin{aligned}
= \quad & \{\ x_s \cup x_t = (s{\cap}t) - u\ \} \\
& (u - (s{\cap}t)) \cup ((s{\cap}t) - u) \\
= \quad & \{\ \text{set calculus}\ \} \\
& u \,\Delta\, (s{\cap}t)
\end{aligned}
$$

while also

$$
\begin{aligned}
& d_s \cap d_t \\
= \quad & \{\ \text{choice of } d_s, d_t\ \} \\
& ((u{-}s) \cup x_s) \cap ((u{-}t) \cup x_t) \\
= \quad & \{\ \text{set calculus}\ \} \\
& (u - (s{\cup}t)) \cup (x_s \cap (u{-}t)) \cup (x_t \cap (u{-}s)) \cup (x_s \cap x_t) \\
= \quad & \{\ x_s \mathrel{\ast\!\!\!|} x_t\ \} \\
& (u - (s{\cup}t)) \cup (x_s \cap (u{-}t)) \cup (x_t \cap (u{-}s)) \\
= \quad & \{\ x_s, x_t \subseteq (s{\cap}t){-}u \mathrel{\ast\!\!\!|} u\ \} \\
& u - (s{\cup}t)
\end{aligned}
$$

We now show that this choice is indeed allowed:

$$
\begin{aligned}
& s' \cap t' \\
= \quad & \{\ \text{definitions of } d_s \text{ and } d_t\ \} \\
& (s\Delta d_s) \cap (t\Delta d_t) \\
= \quad & \{\ \text{choice of } d_s, d_t\ \} \\
& (s\Delta((u{-}s) \cup x_s)) \cap (t\Delta((u{-}t) \cup x_t)) \\
= \quad & \{\ \text{set calculus}\ \} \\
& ((u{\cup}s)\Delta x_s) \cap ((u{\cup}t)\Delta x_t) \\
= \quad & \{\ x_s \subseteq s,\ x_t \subseteq t\ \} \\
& ((u{\cup}s) - x_s) \cap ((u{\cup}t) - x_t)
\end{aligned}
$$

$=$ $\{$ set calculus $\}$

$(u \cup (s \cap t)) - (x_s \cup x_t)$

$=$ $\{ x_s \cup x_t = (s \cap t) - u \}$

$(u \cup (s \cap t)) - ((s \cap t) - u)$

$=$ $\{$ set calculus $\}$

u

We are not yet quite done, since in general there is a range of choices, each giving the same minimal value for $\#d_s + \#d_t$. We break the tie by using the well-order of Section 3.1 on pairs of finite sets, in this case the set of pairs $\langle s', t' \rangle$ consistent with the construction above. This means we have to minimize $\#s'$, and therefore to maximize $\#x_s$. Now the largest possible x_s is the one that takes up the whole "jointly required" set $(s \cap t) - u$, leaving $\emptyset$ for x_t. Then

s'

$=$ $\{$ as above $\}$

$(u \cup s) - x_s$

$=$ $\{$ largest x_s $\}$

$(u \cup s) - ((s \cap t) - u)$

$=$ $\{$ set calculus $\}$

$u \cup (s - t)$

and

t'

$=$ $\{$ as above $\}$

$(u \cup t) - x_t$

$=$ $\{ x_t = \emptyset \}$

$u \cup t$

In the other direction we use the functionality of $\langle\!\langle\cap\rangle\!\rangle$.

As an illustration of the solution found for $\triangleleft$, take

$$\begin{array}{rcl} s &=& \{1,3,4,7\} \\ t &=& \{2,3,4,6\} \\ u &=& \{1,2,4,5\} \end{array}$$

We find

$$\begin{array}{rcl} s' &=& \{1,2,4,5\}\cup(\{1,3,4,7\}-\{2,3,4,6\}) \\ &=& \{1,2,4,5\}\cup\{1,7\} \\ &=& \{1,2,4,5,7\} \\ t' &=& \{1,2,4,5\}\cup\{2,3,4,6\} \\ &=& \{1,2,3,4,5,6\} \end{array}$$

Indeed,

$$\begin{array}{rcl} s'\cap t' &=& \{1,2,4,5,7\}\cap\{1,2,3,4,5,6\} \\ &=& \{1,2,4,5\} \\ &=& u \end{array}$$

5.2.e Union

Relation: $\langle\!\langle\cup\rangle\!\rangle\colon \mathcal{P}A\times\mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\ \langle s,t\rangle\triangleleft u \ = \ \langle u\cap s, u-(s-t)\rangle\ ]$
$[\ \langle s,t\rangle\triangleright u \ = \ s\cup t\ ]$

We use $s\cup t \ = \ \overline{\overline{s}\cap\overline{t}}$, so that we can derive the answer from the solution for intersection. We have to be careful, though, since in the tie breaking we want the smallest possible s, but using the tie breaking for $\cap$ would give the smallest $\overline{s}$, and therefore the largest s. This can be resolved as follows. Since the collection of candidates that tie are pairs of sets all having the same symmetric difference, smaller for one of the two sets means larger for the other one. So, if restricted to pairs from the tying set, the well-order on pairs of sets has the property that

$$[\ \langle\,\overline{s},\overline{t}\,\rangle \preceq \langle\,\overline{s'},\overline{t'}\,\rangle \quad\equiv\quad \langle t,s\rangle \preceq \langle t',s'\rangle\ ]$$

We therefore get the desired tie breaking if we switch the s- and t-positions:

$$
\begin{array}{ll}
 & \langle s, t\rangle \triangleleft_{\cup} u \\
= & \quad \{ \text{ switching } s \text{ and } t \text{ to get the right tie breaking } \} \\
 & \langle s', t'\rangle \ \text{ where } \ \langle \overline{t'}, \overline{s'}\,\rangle = \langle\, \overline{t}, \overline{s}\,\rangle \triangleleft_{\cap} \overline{u} \\
= & \quad \{ \text{ definition of } \triangleleft_{\cap} \ \} \\
 & \langle s', t'\rangle \ \text{ where } \ \langle \overline{t'}, \overline{s'}\,\rangle = \langle \overline{u} \cup (\overline{t} - \overline{s}), \overline{u} \cup \overline{s}\rangle \\
= & \quad \{ \text{ eliminate “where”, set complement is an involution } \} \\
 & \langle\, \overline{\overline{u} \cup \overline{s}}, \overline{\overline{u} \cup (\overline{t} - \overline{s})}\,\rangle \\
= & \quad \{ \text{ set calculus } \} \\
 & \langle u \cap s, u - (s - t)\rangle
\end{array}
$$

As an illustration, take again

$$
\begin{array}{lcl}
s & = & \{1, 3, 4, 7\} \\
t & = & \{2, 3, 4, 6\} \\
u & = & \{1, 2, 4, 5\}
\end{array}
$$

Then

$$
\begin{array}{lcl}
s' & = & \{1, 2, 4, 5\} \cap \{1, 3, 4, 7\} \\
 & = & \{1, 4\} \\
t' & = & \{1, 2, 4, 5\} - (\{1, 3, 4, 7\} - \{2, 3, 4, 6\}) \\
 & = & \{1, 2, 4, 5\} - \{1, 7\} \\
 & = & \{2, 4, 5\}
\end{array}
$$

Indeed,

$$
\begin{array}{lcl}
s' \cup t' & = & \{1, 4\} \cup \{2, 4, 5\} \\
 & = & \{1, 2, 4, 5\} \\
 & = & u
\end{array}
$$

5.2.f Asymmetric set difference

For the asymmetric operation $-$ we treat both $\langle\!\langle - \rangle\!\rangle$ and $\langle\!\langle \dot{-} \rangle\!\rangle$

Relation: $\langle\!\langle - \rangle\!\rangle \colon \mathcal{P}A \times \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\ \langle s,t\rangle \triangleleft u \ = \ \langle u \cup (s \cap t), t - u\rangle\ ]$
$[\ \langle s,t\rangle \triangleright u \ = \ s - t\ ]$

Relation: $\langle\!\langle \breve{-} \rangle\!\rangle \colon \mathcal{P}A \times \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\ \langle s,t\rangle \triangleleft u \ = \ \langle s - u, u \cup (s \cap t\rangle\ ]$
$[\ \langle s,t\rangle \triangleright u \ = \ s - t\ ]$

We obtain $\triangleleft_{-}$ analogously to $\triangleleft_{\cup}$:

$$\begin{array}{ll}
 & \langle s,t\rangle \triangleleft_{-} u \\
= & \quad \{\ s - t \ = \ s \cap \overline{t}\ \} \\
 & \langle s',t'\rangle \ \ \text{where} \ \ \langle\, s', \overline{t'}\,\rangle = \langle\, s, \overline{t}\,\rangle \triangleleft_{\cap} \overline{u} \\
= & \quad \{\ \text{definition of } \triangleleft_{\cap}\ \} \\
 & \langle s',t'\rangle \ \ \text{where} \ \ \langle\, s', \overline{t'}\,\rangle = \langle\, u \cup (s - \overline{t}), u \cup \overline{t}\,\rangle \\
= & \quad \{\ \text{eliminate “where”, set complement is an involution}\ \} \\
 & \langle\, u \cup (s - \overline{t}), \overline{u \cup \overline{t}}\,\rangle \\
= & \quad \{\ \text{set calculus}\ \} \\
 & \langle u \cup (s \cap t), t - u\rangle
\end{array}$$

Again, take

$$\begin{array}{lcl}
s & = & \{1,3,4,7\} \\
t & = & \{2,3,4,6\} \\
u & = & \{1,2,4,5\}
\end{array}$$

Then

$$\begin{array}{lcl}
s' & = & \{1,2,4,5\} \cup (\{1,3,4,7\} \cap \{2,3,4,6\}) \\
 & = & \{1,2,4,5\} \cup \{3,4\} \\
 & = & \{1,2,3,4,5\} \\
t' & = & \{2,3,4,6\} - \{1,2,4,5\} \\
 & = & \{3,6\}
\end{array}$$

Indeed,

$$\begin{aligned} s' - t' &= \{1,2,3,4,5\} - \{3,6\} \\ &= \{1,2,4,5\} \\ &= u \end{aligned}$$

Since tie breaking is asymmetric, it is not obvious that $\lhd_{\underline{\smile}}$ can be found by simply switching the s-field with the t-field in $\lhd_{-}$. As it is, this happens to give the right answer. The explanation is that this switch gives just the right tie breaking, as before for $\lhd_{\cup}$.

5.2.g Symmetric set difference

Relation: $\langle\!\langle \Delta \rangle\!\rangle \colon \mathcal{P}A \times \mathcal{P}A \sim \mathcal{P}A$

Maintainer: $[\ \langle s,t\rangle \lhd u \ = \ \langle (s{\cap}t)\Delta(s{\cap}u), (s{\cap}t) \cup (u{-}s)\rangle\ ]$
$[\ \langle s,t\rangle \rhd u \ = \ s\Delta t\ ]$

Putting again $\langle d_s, d_t\rangle = \langle s\Delta s', t\Delta t'\rangle$, where we want to minimize $\#d_s + \#d_t$, the requirement is that

$$\begin{aligned} & u \ = \ s' \,\Delta\, t' \\ = \quad & \{ \text{ definitions of } d_s \text{ and } d_t\text{; properties of } \Delta \ \} \\ & u \ = \ s \,\Delta\, t \,\Delta\, d_s \,\Delta\, d_t \\ = \quad & \{\ \Delta \text{ is involutive } \} \\ & d_s \,\Delta\, d_t \ = \ s \,\Delta\, t \,\Delta\, u \end{aligned}$$

Since $\#d_s + \#d_t \ = \ \#(d_s \Delta d_t) + 2\#(d_s \cap d_t) \ = \ \#(s\Delta t\Delta u) + 2\#(d_s{\cap}d_t)$, we want to minimize $\#(d_s{\cap}d_t)$, which certainly succeeds if we have $d_s \nmid d_t$. We calculate:

$$\begin{aligned} & d_s \nmid d_t \\ \equiv \quad & \{ \text{ definition of } \nmid \ \} \\ & d_s \cap d_t \ = \ \emptyset \end{aligned}$$

$\equiv \qquad \{\ d_s \,\Delta\, d_t \ = \ s \,\Delta\, t \,\Delta\, u\ \}$

$d_s \cap (d_s \,\Delta\, s \,\Delta\, t \,\Delta\, u) \ = \ \emptyset$

$\equiv \qquad \{\ \cap$ distributes over $\Delta\ \}$

$d_s \,\Delta\, (d_s \cap (s \Delta t \Delta u)) \ = \ \emptyset$

$\equiv \qquad \{\ \Delta$ is involutive $\}$

$d_s \ = \ d_s \cap (s \Delta t \Delta u)$

$\equiv \qquad \{$ set calculus $\}$

$d_s \subseteq s \,\Delta\, t \,\Delta\, u$

This inequation has possibly many solutions in d_s. To break the tie, we additionally minimize $\#s'$. For this we find:

$\#s'$

$= \qquad \{$ definition of $d_s\ \}$

$\#(s \Delta d_s)$

$= \qquad \{$ splitting d_s into $d_s \cap s$ and $d_s - s\ \}$

$\#s - \#(d_s \cap s) + \#(d_s - s)$

Clearly, we must keep $d_s \subseteq s$, and further as large as possible, which gives us

d_s

$= \qquad \{$ combining the two upper bounds $\}$

$s \cap (s \Delta t \Delta u)$

$= \qquad \{\ \cap$ distributes over $\Delta\ \}$

$s \,\Delta\, (s \cap (t \Delta u))$

Now

s'

$$\begin{array}{cl}
= & \quad \{ \text{ definition of } d_s \ \} \\
 & s \,\Delta\, d_s \\
= & \quad \{ \text{ solution found for } d_s \ \} \\
 & s \,\Delta\, s \,\Delta\, (s \cap (t \Delta u)) \\
= & \quad \{ \ \Delta \text{ is involutive } \} \\
 & s \cap (t \Delta u) \\
= & \quad \{ \ \cap \text{ distributes over } \Delta \ \} \\
 & (s{\cap}t) \,\Delta\, (s{\cap}u)
\end{array}$$

and

$$\begin{array}{cl}
 & t' \\
= & \quad \{ \ s' \,\Delta\, t' = u \ \} \\
 & s' \,\Delta\, u \\
= & \quad \{ \text{ solution found for } s' \ \} \\
 & (s{\cap}t) \,\Delta\, (s{\cap}u) \,\Delta\, u \\
= & \quad \{ \ \cap \text{ distributes over } \Delta \ \} \\
 & (s{\cap}t) \,\Delta\, ((s \Delta u) \cap u) \\
= & \quad \{ \ - \text{ expressed in } \Delta \ \} \\
 & (s{\cap}t) \,\Delta\, (u{-}s) \\
= & \quad \{ \ s{\cap}t \nmid u - s \ \} \\
 & (s{\cap}t) \cup (u{-}s)
\end{array}$$

For the running example of $s = \{1, 3, 4, 7\}, t = \{2, 3, 4, 6\}, u = \{1, 2, 4, 5\}$, we compute:

$$\begin{array}{rcl}
s' & = & (\{1,3,4,7\}{\cap}\{2,3,4,6\}) \,\Delta\, (\{1,3,4,7\}{\cap}\{1,2,4,5\}) \\
 & = & \{3,4\} \,\Delta\, \{1,4\} \\
 & = & \{1,3\} \\
t' & = & (\{1,3,4,7\}{\cap}\{2,3,4,6\}) \cup (\{1,2,4,5\}{-}\{1,3,4,7\}) \\
 & = & \{3,4\} \cup \{2,5\} \\
 & = & \{2,3,4,5\}
\end{array}$$

Indeed,

$$\begin{aligned} s' \mathbin{\Delta} t' &= \{1,3\} \mathbin{\Delta} \{2,3,4,5\} \\ &= \{1,2,4,5\} \\ &= u \end{aligned}$$

5.3 Constraints involving lists

A constraint between two data structures may bring about a notion of *correspondence* between certain entries of the two data structures. In trying to apply the Principle of Least Change, not only do we want to keep the edit distance small, but — as least as importantly — we also want to keep correspondences intact as much as possible. To express that with the kind of notions introduced thus far is not directly formally possible, since it involves something (namely the notion of “correspondence”) that is extraneous to the choice space. It could be modelled using the “standard decomposition” of Section 4.1. However, here we shall use a less formal and more operational approach.

We introduce a new notion here, namely that of *edit sequences*. The idea is that a change to a data structure can be viewed as the result of a sequence of “elementary” changes, or *edit operations*. The advantage of this viewpoint is that it makes explicit which entries are not involved in a change. The edit operations are the constructors of an algebra for the data type, where the algebra need not be free but may satisfy a number of laws. We could already have taken that viewpoint for finite sets. If the edit operations on sets consist of the insertion or removal of a *single* element at a time, any change can be described as a sequence of edit operations. In fact, usually there are many ways. Some of those involve redundant operations, such as adding an element and removing it later in the edit sequence. An edit sequence not containing such redundant operations is called *irreducible*. For sets it is not hard to show that for the constraints dealt with in Section 5.2, with the notable exception of $\in$, all irreducible edit sequences yield the same result as the one-fell-swoop approach of that section. We call that property of a semi-maintainer (which is relative to the set of edit operations) *indifference*. Any semi-maintainer of a functional relation in the direction argument$\rightarrow$result, is, of course, indifferent, regardless of what edit operations are included.

The following example shows that $\lhd_\in$ is not indifferent. Assume that in the constraint $x \in s$ initially $x = 3$ and $s = \{0,2,3,5\}$. We consider two irreducible edit sequences resulting in $s'' = \{0,5\}$, namely

$$\begin{array}{llllll}
\text{(a)} & s = \{0,2,3,5\} & \rightsquigarrow & s' = \{0,3,5\} & \rightsquigarrow & s'' = \{0,5\} \\
\text{(b)} & s = \{0,2,3,5\} & \rightsquigarrow & s' = \{0,2,5\} & \rightsquigarrow & s'' = \{0,5\}
\end{array}$$

With route (a) we get $x' = 3\S\{0,3,5\} = 3$, $x'' = 3\S\{0,5\} = 5$. With route (b) we get $x' = 3\S\{0,2,5\} = 2$, $x'' = 2\S\{0,5\} = 0$.

The type of finite lists of A-elements is denoted as $List(A)$; for the type of non-empty lists we use $List_+(A)$. For lists, the simplest set of edit operations is:

$$\begin{array}{ll}
\mathsf{nil}: & 1 \qquad\qquad \to List\,(A) \\
\quad \text{— create an empty list} & \\
\mathsf{cons}\langle a,x\rangle: & A \times List(A) \to List_+(A) \\
\quad \text{— prepend } a \text{ to list } x &
\end{array}$$

Instead of nil we also write [ ], and for $\mathsf{cons}\langle a,x\rangle$ we also write $a{:}x$. So $3 : [1,4] = [3,1,4]$. For denoting the result of concatenating two lists, we use $+\!\!+$: $[3,1]+\!\!+[4,1,5,9] = [3,1,4,1,5,9]$.

The corresponding *destructor* functions head: $List_+(A) \to A$ and tail : $List_+(A) \to List(A)$ are characterized by:

$$[\,\mathsf{cons}\langle a,x\rangle = y \quad\equiv\quad \langle a,x\rangle = \langle \mathsf{head}(y), \mathsf{tail}(y)\rangle\,]$$

More advanced edit operations are:

$$\begin{array}{ll}
\mathsf{insert}\langle x,i,a\rangle: & List\,(A) \times \mathbb{N} \times A \to List_+(A) \\
\quad \text{— insert a new list entry } a \text{ at position } i & \\
\mathsf{delete}\langle x,i\rangle: & List_+(A) \times \mathbb{N} \qquad \to List\,(A) \\
\quad \text{— delete an existing list entry} & \\
\mathsf{change}\langle x,i,a\rangle: & List_+(A) \times \mathbb{N} \times A \to List_+(A) \\
\quad \text{— change the value of an existing list entry} &
\end{array}$$

The index arguments i must be in range. The effect of insertion is that entries with higher indices are moved up:

$$\mathsf{insert}\langle[6,1,5,7],2,4\rangle = [6,1,4,5,7]$$

We have:

$$\begin{array}{lll}
[\ \mathsf{insert}\langle x,0,b\rangle & = & b : x\ ] \\
[\ \mathsf{insert}\langle a{:}x,i{+}1,b\rangle & = & a : \mathsf{insert}\langle x,i,b\rangle\ ] \\
[\ \mathsf{delete}\langle a{:}x,0\rangle & = & x\ ] \\
[\ \mathsf{delete}\langle a{:}x,i{+}1\rangle & = & a : \mathsf{delete}\langle x,i\rangle\ ] \\
[\ \mathsf{change}\langle a{:}x,0,b\rangle & = & b : x\ ] \\
[\ \mathsf{change}\langle a{:}x,i{+}1,b\rangle & = & a : \mathsf{change}\langle x,i,b\rangle\ ]
\end{array}$$

The destructor functions $\Box[i]\colon List(A) \to A$, where the index i must be in range, are characterized by:

$$[\ (\mathsf{insert}\langle x,i,a\rangle)[i] = a\ ]$$

We use, without further justification, the usual laws for these operations, such as $[\ \mathsf{change}\langle\mathsf{insert}\langle x,i,a\rangle,i,b\rangle\ =\ \mathsf{insert}\langle x,i,b\rangle\ ]$, which can be derived from the other equations.

The set of edit operations insert, delete and change is expressible in terms of the one-element set consisting of a replace operation that allows replacing an arbitrary segment of a list of length m by a list of length n. We have:

$$\begin{array}{lll}
[\ \mathsf{replace}\langle x,0,0,y\rangle & = & y +\!\!+\, x\ ] \\
[\ \mathsf{replace}\langle a{:}x,0,m{+}1,y\rangle & = & \mathsf{replace}\langle x,0,m,y\rangle\ ] \\
[\ \mathsf{replace}\langle a{:}x,i{+}1,m,y\rangle & = & a : \mathsf{replace}\langle x,i,m,y\rangle\ ]
\end{array}$$

$$\begin{array}{lll}
[\ \mathsf{insert}\langle x,i,a\rangle & = & \mathsf{replace}\langle x,i,0,[a]\rangle\ ] \\
[\ \mathsf{delete}\langle x,i\rangle & = & \mathsf{replace}\langle x,i,0,[\,]\rangle\ ] \\
[\ \mathsf{change}\langle x,i,a\rangle & = & \mathsf{replace}\langle x,i,1,[a]\rangle\ ]
\end{array}$$

5.3.a Cons, head, tail

Relation: $\langle\!\langle \mathsf{cons} \rangle\!\rangle \colon A \times List(A) \sim List_{+}(A)$

Maintainer:
$$\begin{aligned}
&[\ \langle a, x\rangle \lhd y \ = \ \langle \mathsf{head}(y), \mathsf{tail}(y)\rangle\ ]\\
&[\ \langle a, x\rangle \rhd y \ = \ \mathsf{cons}\langle a, x\rangle\ ]
\end{aligned}$$

Relation: $\langle\!\langle \mathsf{head} \rangle\!\rangle \colon List_{+}(A) \sim A$

Maintainer:
$$\begin{aligned}
&[\ x \lhd a \ = \ \mathsf{cons}\langle a, \mathsf{tail}(x)\rangle\ ]\\
&[\ x \rhd a \ = \ \mathsf{head}(x)\ ]
\end{aligned}$$

Relation: $\langle\!\langle \mathsf{tail} \rangle\!\rangle \colon List_{+}(A) \sim List(A)$

Maintainer:
$$\begin{aligned}
&[\ x \lhd y \ = \ \mathsf{cons}\langle \mathsf{head}(x), y\rangle\ ]\\
&[\ x \rhd y \ = \ \mathsf{tail}(x)\ ]
\end{aligned}$$

From the characterization of head and tail,

$$\mathsf{cons}\langle a, x\rangle = y \quad \equiv \quad \langle a, x\rangle = \langle \mathsf{head}(y), \mathsf{tail}(y)\rangle$$

we see that cons is a bijection, meaning that $\langle\!\langle \mathsf{cons} \rangle\!\rangle$ is functional in both directions. Its maintainer now follows immediately.

We further see that $\langle\!\langle \mathsf{head} \rangle\!\rangle = (\langle\!\langle \mathsf{cons} \rangle\!\rangle^{\smile} ; \langle\!\langle \pi_L \rangle\!\rangle)$ and $\langle\!\langle \mathsf{tail} \rangle\!\rangle = (\langle\!\langle \mathsf{cons} \rangle\!\rangle^{\smile} ; \langle\!\langle \pi_R \rangle\!\rangle)$. Duality, in combination with Theorem 12, now gives the solutions.

5.3.b Map

Relation: $\mathsf{map}(R) \colon List(A) \sim List(B)$ where $R \colon A \sim B$

The relation $x(\mathsf{map}(R))y$ holds whenever x and y have the same set of indices, and $[\ (x[i])\, R\, (y[i])\ ]$. We assume the existence of a maintainer of R.

Maintainer:
$$\begin{aligned}
&[\ \mathsf{nil} \rhd y &&= \ \mathsf{nil}\ ]\\
&[\ \mathsf{insert}\langle x, i, a\rangle \rhd y &&= \ \mathsf{insert}\langle y, i, a \rhd_R \mathsf{init}(B)\rangle\ ]\\
&[\ \mathsf{delete}\langle x, i\rangle \rhd y &&= \ \mathsf{delete}\langle y, i\rangle\ ]\\
&[\ \mathsf{change}\langle x, i, a\rangle \rhd y &&= \ \mathsf{change}\langle y, i, a \rhd_R y[i]\rangle\ ]
\end{aligned}$$

We treat only the semi-maintainer $\triangleright$ because the other one is dual, using that converse commutes with map.

The semi-maintainer given above is obvious — there is no other choice — except for the occurrence of $\mathsf{init}(B)$. The explanation is that we view insert as the combination of a virtual create operation *immediately* followed by a mandatory change operation:

$$\mathsf{insert}\langle x, i, a\rangle \;=\; \mathsf{change}\langle \mathsf{create}\langle x, i\rangle, a\rangle$$

The result of $\mathsf{create}\langle x, i\rangle$ on a list x of type $List(A)$ is the same as that of $\mathsf{insert}\langle x, i, \mathsf{init}(A)\rangle$, and its contribution to $\triangleright$ is:

$$[\ \mathsf{create}\langle x, i\rangle \triangleright y \;=\; \mathsf{create}\langle y, i\rangle\ ]$$

So, putting $y' = \mathsf{create}\langle x, i\rangle \triangleright y$ and $y'' = \mathsf{change}\langle x', i, a\rangle \triangleright y'$, where $x' = \mathsf{create}\langle x, i\rangle$, we have

$$\begin{array}{ll}
 & y'[i] \\
= & \quad \{\ \text{definition of } y'\ \} \\
 & (\mathsf{create}\langle x, i\rangle \triangleright y)[i] \\
= & \quad \{\ \triangleright \text{ for } \mathsf{create}\ \} \\
 & (\mathsf{create}\langle y, i\rangle)[i] \\
= & \quad \{\ \text{result of } \mathsf{create}\ \} \\
 & (\mathsf{insert}\langle y, i, \mathsf{init}(B)\rangle)[i] \\
= & \quad \{\ \mathsf{insert}\text{-indexing law}\ \} \\
 & \mathsf{init}(B)
\end{array}$$

so that

$$\begin{array}{ll}
 & y'' \\
= & \quad \{\ \text{definition of } y''\ \}
\end{array}$$

$\mathsf{change}\langle x', i, a\rangle \triangleright y'$

$= \quad \{ \triangleright \text{ for } \mathsf{change} \}$

$\mathsf{change}\langle y', i, a \triangleright_R y'[i]\rangle$

$= \quad \{ \text{ above result } \}$

$\mathsf{change}\langle y', i, a \triangleright_R \mathsf{init}(B)\rangle$

$= \quad \{ \text{ definition of } y'\text{, result of } \mathsf{create} \}$

$\mathsf{change}\langle \mathsf{insert}\langle y, i, \mathsf{init}(B)\rangle, i, a \triangleright_R \mathsf{init}(B)\rangle$

$= \quad \{ \mathsf{change\text{-}insert} \text{ law } \}$

$\mathsf{insert}\langle y, i, a \triangleright_R \mathsf{init}(B)\rangle$

In an implementation, $\mathsf{map}(R)$ maintenance can be realized by setting up a (trivial) network of R constraints between corresponding list entries, as suggested in Figure 5. This network is modified in the obvious way for each structural change of the lists involved.

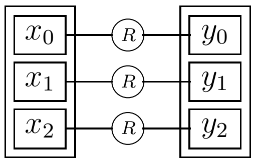

Figure 5: The network created for constraint $\mathsf{map}(R)$ between to three-element lists.

5.3.c Fold

Relation: $\mathsf{fold}(R)$: $List_+(A) \sim A$ where R: $A \times A \sim A$

Maintainer:
$[\ \mathsf{insert}\langle x, i, a\rangle \triangleright y = \mathsf{insert}\langle y, i, a \triangleright_R \mathsf{init}(B)\rangle\ ]$
$[\ \mathsf{delete}\langle x, i\rangle \triangleright y = \mathsf{delete}\langle y, i\rangle\ ]$
$[\ \mathsf{change}\langle x, i, a\rangle \triangleright y = \mathsf{change}\langle y, i, a \triangleright_R y[i]\rangle\ ]$

5.4 Constraints involving numbers

In the following $\mathbb{R}$ stands for the set of real numbers and $\mathbb{Z}$ for the set of integers. We use $\mathbb{V}$ to stand for either of $\mathbb{R}$ or $\mathbb{Z}$, with the convention that in any paragraph the choice for either reading is made consistently for the scope of that paragraph. For real numbers we make use of the assumption that the set of representable numbers (for example, floating-point numbers) is nowhere dense, and in particular that for any $b \in \mathbb{R}$ there is a largest value $\beta \in \mathbb{R}$ such that $\beta < b$, so $[\, x < b \equiv x \leq \beta \,]$. This number β will be denoted by $b{-}\varepsilon$.

For intervals of numbers we have the following shorthands:

$$\begin{array}{lcll}
\mathbb{V}[a^\frown b] & = & (x : x \in \mathbb{V} : a \leq x \leq b) & \text{, for } a, b \in \mathbb{V}, a \leq b \\
\mathbb{V}[a^\frown\ ) & = & (x : x \in \mathbb{V} : a \leq x) & \text{, for } a \in \mathbb{V} \\
\mathbb{V}(\ ^\frown b] & = & (x : x \in \mathbb{V} : x \leq b) & \text{, for } b \in \mathbb{V} \\
\mathbb{V}[a^\frown b) & = & (x : x \in \mathbb{V} : a \leq x < b) & \text{, for } a, b \in \mathbb{V}, a < b
\end{array}$$

Note that $\mathbb{R}[a^\frown b) = \mathbb{R}[a^\frown b{-}\varepsilon]$, and $\mathbb{Z}[a^\frown b) = \mathbb{Z}[a^\frown b{-}1]$.

For numbers there is an obvious notion of closeness: x is closer to a than x' whenever $|x - a| < |x' - a|$. With this interpretation we have:

$$\begin{array}{lcl}
[\, x \S \mathbb{V}[a^\frown b] & = & a \uparrow x \downarrow b \,] \\
[\, x \S \mathbb{V}[a^\frown\ ) & = & a \uparrow x \,] \\
[\, x \S \mathbb{V}(\ ^\frown b] & = & x \downarrow b \,] \\
[\, x \S \mathbb{R}[a^\frown b) & = & a \uparrow x \downarrow (b{-}\varepsilon) \,] \\
[\, x \S \mathbb{Z}[a^\frown b) & = & a \uparrow x \downarrow (b{-}1) \,]
\end{array}$$

In the right-hand sides of these equalities parentheses are not needed to indicate which of $\uparrow$ and $\downarrow$ takes precedence, since, for $a \leq b$, $(a \uparrow x) \downarrow b = a \uparrow (x \downarrow b)$.

5.4.a Unequal

Relation: $\neq : \mathbb{Z} \sim \mathbb{Z}$

$$\begin{array}{lrcll}
\text{Maintainer:} & [\,x\triangleleft y & = & x & \text{if } x \neq y, \\
 & & = & x-1 & \text{if } x = y \,\wedge\, x > 0, \\
 & & = & x+1 & \text{if } x = y \,\wedge\, x \leq 0\,] \\[1ex]
 & [\,x\triangleright y & = & y\triangleleft x\,] &
\end{array}$$

This was treated as an example in Section 4.2.

5.4.b At most

Relation: $\leq\colon \mathbb{V} \sim \mathbb{V}$

$$\begin{array}{lrcl}
\text{Maintainer:} & [\,x\triangleleft y & = & x \downarrow y\,] \\
 & [\,x\triangleright y & = & x \uparrow y\,]
\end{array}$$

The constraint $x \leq y$ can be expressed equivalently both as $x \in \mathbb{V}(\,^\frown y]$ and as $y \in \mathbb{V}[x^\frown\,)$. From this the result is immediate.

5.4.c Absolute value

Relation: $\langle\!\langle\, |\square|\, \rangle\!\rangle\colon \mathbb{V} \sim \mathbb{V}[0^\frown\,)$

$$\begin{array}{lrcrl}
\text{Maintainer:} & [\,x\triangleleft y & = & y & \text{if } x \geq 0, \\
 & & = & -y & \text{if } x < 0\,] \\[1ex]
 & [\,x\triangleright y & = & |x|\,] &
\end{array}$$

The constraint $|x| = y$ can be expressed equivalently as $x \in \{-y, y\}$, giving $[\,x\triangleleft y \;=\; x\S\{-y, y\}\,]$. If the old value of x is 0 and the new value of y is not 0, the selection from $\{-y, y\}$ requires a tie to be broken, and it is suggested to pick then the positive candidate, y. Otherwise, the candidate closest to x has to be picked, which depends purely on the sign of x.

5.4.d Floor

Relation: $\langle\!\langle \mathsf{floor} \rangle\!\rangle\colon \mathbb{R} \sim \mathbb{Z}$

Maintainer: $[\, x \triangleleft n \;=\; n \downarrow x \uparrow (n+1) - \varepsilon \,]$
$[\, x \triangleright n \;=\; \mathsf{floor}(x) \,]$

The constraint $x \langle\!\langle \mathsf{floor} \rangle\!\rangle n$ can be expressed equivalently as $x \in \mathbb{R}[n \frown n+1)$.

5.4.e Modulo

Relation: $\langle\!\langle \mathsf{mod}\ a \rangle\!\rangle \colon \mathbb{V} \sim \mathbb{V}[0 \frown a)$

Maintainer: $[\, x \triangleleft y \;=\; y + a \times \mathsf{round}((x-y)/a) \,]$
$[\, x \triangleright y \;=\; x\ \mathsf{mod}\ a \,]$

Here the parameter a is assumed to be a positive (constant) number in $\mathbb{V}$. The constraint $x \langle\!\langle \mathsf{mod}\ a \rangle\!\rangle y$ can be expressed equivalently as $x \in y + a \times \mathbb{Z}$, in which the operations $a\times$ and $y+$ have been extended to sets. The problem is thus to find a nice expression for $x \S (y + a \times \mathbb{Z})$. The linear bijection f defined by $[\, f(\xi) = y + a \times \xi \,]$ is monotonic in the order induced by closeness. So $x \S (y + a \times \mathbb{Z}) = f(f^{-1}(x) \S \mathbb{Z})$. Now, in general, $\xi \S \mathbb{Z} = \mathsf{round}(\xi)$, which gives us the maintainer.

It is instructive to see what happens, for $a = 100$, say, when the value of y is 99 and is 'incremented' by 1 modulo 100, thus jumping back to 00. Assume, for the sake of concreteness, that the old value of x is 1999. Then the new value becomes

$$\begin{array}{ll} & 1999 \triangleleft 0 \\ = & \quad \{ \text{ definition of } \triangleleft \text{ } \} \\ & 0 + 100 \times \mathsf{round}((1999-0)/100) \\ = & \quad \{ \text{ arithmetic } \} \\ & 100 \times \mathsf{round}(19.99) \\ = & \quad \{ \text{ round, arithmetic } \} \\ & 2000 \end{array}$$

Rounding may require tie breaking, as when (still for $a = 100$) the value of x is 1992, and the value of y jumps from 92, say, to 42. The two candidates

1942 and 2042 are equally close to 1992. The tie-breaking method suggested earlier picks 1942 as being the value closest to 0.

5.4.f Polar to Cartesian

Relation: $\langle\!\langle \mathsf{pol2cart} \rangle\!\rangle\colon \mathbb{R}[0\frown) \times \mathbb{R} \sim \mathbb{R} \times \mathbb{R}$

Maintainer:

$$\begin{array}{rcll} [\ \langle r,\varphi\rangle \lhd \langle x,y\rangle & = & \langle 0,\varphi\rangle & \text{if } \langle x,y\rangle = \langle 0,0\rangle, \\ & = & \langle z, \alpha + 2\pi \times \mathsf{round}((\varphi-\alpha)/2\pi)\rangle & \text{otherwise,} \\ & & \text{where } z = \mathsf{radius}\langle x,y\rangle & \\ & & \qquad\ \alpha = \mathsf{angle}\langle x,y\rangle\ ] & \end{array}$$

$$[\ \langle r,\varphi\rangle \rhd \langle x,y\rangle \ = \ (r \times \mathsf{cos}(\varphi), r \times \mathsf{sin}(\varphi))\ ]$$

Here the function $\mathsf{pol2cart}$ is defined by

$$[\ \mathsf{pol2cart}\langle r,\varphi\rangle = (r \times \mathsf{cos}(\varphi), r \times \mathsf{sin}(\varphi))\ ]$$

The constraint $\mathsf{pol2cart}\langle r,\varphi\rangle = \langle x,y\rangle$ can be expressed equivalently as

$$r = \mathsf{radius}\langle x,y\rangle \ \wedge\ (\langle x,y\rangle = \langle 0,0\rangle \vee \varphi \ \mathsf{mod}\ 2\pi = \mathsf{angle}\langle x,y\rangle)$$

where the function angle is partial: it is defined on $\mathbb{R} \times \mathbb{R}$ with the exception of the origin $\langle 0,0\rangle$. The choice space is clearly of a 'rectangular' shape $s \times t$, in which s is even a singleton set, since the r component of the polar representation of a point $\langle x,y\rangle$ depends functionally on $\langle x,y\rangle$. We focus therefore on the second conjunct, which gives the φ part of the constraint. We distinguish between the case $\langle x,y\rangle = \langle 0,0\rangle$ and the case $\langle x,y\rangle \neq \langle 0,0\rangle$.

If $\langle x,y\rangle = \langle 0,0\rangle$, the constraint is vacuously satisfied; the choice space for φ is all of $\mathbb{R}$, and we use $\varphi \S \mathbb{R} = \varphi$.

If $\langle x,y\rangle \neq \langle 0,0\rangle$, we can use the solution given above for maintaining $\langle\!\langle \mathsf{mod}\ a \rangle\!\rangle$.

6 More . . .

Hysteresis Example; abs

(semi-)lattices

Normalization

Error handling

References

[1] A. Borning. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM TOPLAS*, 3(4):353–387, October 1981.

[2] Luca Cardelli. Building user interfaces by direct manipulation. In *Proc. ACM SIGGRAPH Symposium on User Interface Software*, pages 152–166. ACM, 1988. Also DEC SRC Research Report 22, 1987; `ftp://ftp.digital.com/pub/DEC/SRC/research-reports/SRC-022.ps.Z`.

[3] Scott Hudson and Roger King. A generator of direct manipulation office systems. *ACM TOPLAS*, 4(2):132–136, April 1986.

[4] Scott Hudson and Roger King. Semantic feedback in the Higgens UIMS. *IEEE Transactions on Software Engineering*, 14(8):1188–1206, August 1988.

[5] G.E. Krasner and S.T. Pope. A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80. *Journal of Object Oriented Programming*, 1(3):26–49, August/September 1988.

[6] Lambert Meertens and Steven Pemberton. The ergonomics of computer interfaces — designing a system for human use. Technical Report CS-R9258, CWI, Amsterdam, 1992. `www.cwi.nl/ftp/CWIreports/AA/CS-R9258.ps.Z`.

[7] Brad A. Myers, Dario A. Giuse, Roger B. Dannenberg, Brad Vander Zanden, David S. Kosbie, Edward Pervin, Andrew Mickish, and Philippe Marchal. Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer*, 23(11):71–85, November 1990. `www.cs.cmu.edu/afs/cs/project/garnet/doc/papers/garnetIEEE.ps`.

[8] Steven Pemberton. The Views application environment. Technical Report CS-R9257, CWI, Amsterdam, 1992. `www.cwi.nl/ftp/CWIreports/AA/CS-R9257.ps.Z`.