Perfect Derived Propagators

Christian Schulte[1] and Guido Tack[2]

[1] ICT, KTH - Royal Institute of Technology, Sweden, `cschulte@kth.se`
[2] PS Lab, Saarland University, Saarbrücken, Germany, `tack@ps.uni-sb.de`

Abstract. When implementing a propagator for a constraint, one must decide about variants: When implementing min, should one also implement max? Should one implement linear equations both with and without coefficients? Constraint variants are ubiquitous: implementing them requires considerable effort, but yields better performance.
This paper shows how to use variable views to derive *perfect* propagator variants: derived propagators inherit essential properties such as correctness and domain and bounds completeness.

1 Introduction

When implementing a propagator for a constraint, one typically needs to decide whether to also implement some of its variants. For example, when implementing a propagator for $\max_{i=1}^{n} x_i = y$, should one also implement $\min_{i=1}^{n} x_i = y$? When implementing the linear equation $\sum_{i=1}^{n} a_i x_i = c$ for integer variables x_i and integers a_i and c, should one implement $\sum_{i=1}^{n} x_i = c$ for better performance?

While resulting in better performance, special implementations for propagator variants inflate code and documentation, and impair maintainability. The approach we take is to derive propagators from already existing propagators using variable views. In [8], we introduced an implementation architecture for variable views to reuse generic propagators without performance penalty. Gecode [4] makes massive use of views: every propagator implementation is reused 3.6 times on average. Without views, Gecode would feature 140 000 rather than 40 000 lines of propagator implementation to be written, tested, and maintained. Due to the extensive use of views, it is vital to develop a model that allows us to prove that derived propagators have the desired properties.

In this paper, we argue that propagators that are derived using variable views are indeed *perfect*: they are not only perfect for performance, we prove that they inherit all essential properties such as correctness and completeness from their original propagator. The key contribution is the identification of properties of views that yield perfect derived propagators. The paper establishes a formal model that defines a view as a function and a derived propagator as functional composition of views (mapping values to values) with a propagator (mapping domains to domains). This model yields all the desired results, in that derived propagators are indeed propagators; faithfully implement the intended constraints; preserve domain completeness of the original propagators; and preserve bounds completeness if the views satisfy additional properties.

2 Preliminaries

We assume a finite set of variables $\mathit{Var} = \{x_1, \ldots, x_n\}$ and a finite set of values Val. An assignment $a \in \mathit{Asn}$ maps variables to values: $\mathit{Asn} = \mathit{Var} \to \mathit{Val}$. A constraint $c \in \mathit{Con}$ is a relation over the variables, represented as the set of all assignments that satisfy the constraint, $\mathit{Con} = 2^{\mathit{Asn}}$.

Constraints are implemented by propagators over domains, which are constructed as follows. A *domain* $d \in \mathit{Dom}$ maps each variable to a finite set of possible values, the *variable domain* $d(x) \subseteq \mathit{Val}$. We identify a domain d with a set of assignments $d \in 2^{\mathit{Asn}}$, and therefore treat domains as constraints to simplify presentation. A domain d_1 is *stronger* than a domain d_2 (written $d_1 \subseteq d_2$), iff for all variables x, $d_1(x) \subseteq d_2(x)$.

Propagators, also called narrowing operators or filter functions, serve as implementations of constraints. A propagator is a function $p \in \mathit{Dom} \to \mathit{Dom}$ that is contracting ($p(d) \subseteq d$) and monotone ($d' \subseteq d \Rightarrow p(d') \subseteq p(d)$).

A propagator p implements its *associated constraint* $c_p = \{a \in \mathit{Asn} \mid p(\{a\}) = \{a\}\}$. Monotonicity implies that $c_p \cap d \subseteq p(d)$ for any domain d: no solution of c_p is ever removed by p. We say that p is *sound* for any $c \subseteq c_p$ and *weakly complete* for any $c' \supseteq c_p$ (it accepts all assignments in c and rejects all assignments not in c'). For any constraint c, we can find a propagator p such that $c = c_p$. Typically, there are several propagators, differing by *propagation strength* (see Sect. 3). Our definitions of soundness and different notions of completeness are based on and equivalent to Benhamou's [1] and Maher's [5].

3 Views and Derived Propagators

A *view* on a variable x is an injective function $\varphi_x \in \mathit{Val} \to \mathit{Val}'$, mapping values from Val to values from a possibly different set Val'. We lift a family of views φ_x point-wise to assignments as follows: $\varphi_{\mathit{Asn}}(a)(x) = \varphi_x(a(x))$. Finally, given a family of views lifted to assignments, we define a view $\varphi \in \mathit{Con} \to \mathit{Con}$ on constraints as $\varphi(c) = \{\varphi_{\mathit{Asn}}(a) \mid a \in c\}$. The inverse of that view is defined as $\varphi^-(c) = \{a \in \mathit{Asn} \mid \varphi_{\mathit{Asn}}(a) \in c\}$. Views can now be composed with a propagator: a *derived propagator* is defined as $\widehat{\varphi}(p) = \varphi^- \circ p \circ \varphi$.

Example. Given a propagator p for the constraint $c \equiv (x = y)$, we want to derive a propagator for $c' \equiv (x = 2y)$ using a view φ such that $\varphi^-(c) = c'$. It is usually easier to think about the other direction: $\varphi(c') \subseteq c$. Intuitively, the function φ leaves x as it is and scales y by 2, while φ^- does the inverse transformation. We thus define $\varphi_x(v) = v$ and $\varphi_y(v) = 2v$. We have a subset relation because some tuples of c may be ruled out by φ.

This example makes clear why the set Val' is allowed to differ from Val. In this particular case, Val' has to contain all multiples of 2 of elements in Val.

The derived propagator is $\widehat{\varphi}(p) = \varphi^- \circ p \circ \varphi$. We say that $\widehat{\varphi}(p)$ "uses a scale view on" y, meaning that φ_y is the function defined as $\varphi_y(v) = 2v$. Similarly, using an identity view on x amounts to φ_x being the identity function on Val.

Given the assignment $a = (x \mapsto 2, y \mapsto 1)$, we first apply φ_{Asn} and get $\varphi_{Asn}(a) = (x \mapsto 2, y \mapsto 2)$. This is returned unchanged by p, so φ^{-} transforms it back to a. Another assignment, $a' = (x \mapsto 1, y \mapsto 2)$, is transformed to $\varphi_{Asn}(a') = (x \mapsto 1, y \mapsto 4)$, rejected ($p(\{\varphi_{Asn}(a')\}) = \emptyset$), and the empty domain is mapped to the empty domain by φ^{-}. Thus, $\widehat{\varphi}(p)$ implements $\varphi^{-}(c)$.

Common views capture linear transformations for integer variables, negation for Boolean variables, or complement for set variables. For example, in [8] the following views are introduced for a variable x and values v: a *minus view* on x is defined as $\varphi_x(v) = -v$, an *offset view* for $o \in \mathbb{Z}$ on x is defined as $\varphi_x(v) = v + o$, and a *scale view* for $a \in \mathbb{Z}$ on x is defined as $\varphi_x(v) = a \cdot v$.

The central properties of derived propagators are expressed in the following theorems (with proofs in the long version of this paper [9]):

Theorem 1. A derived propagator is a propagator: for all propagators p and views φ, $\widehat{\varphi}(p)$ is a monotone and contracting function in $\mathit{Dom} \to \mathit{Dom}$.

Theorem 2. If p implements c_p, then $\widehat{\varphi}(p)$ implements $\varphi^{-}(c_p)$.

Theorem 3. Views preserve contraction: for any domain d, if $p(\varphi(d)) \subsetneq \varphi(d)$, then $\widehat{\varphi}(p)(d) \subset d$. This property makes sure that if the propagator makes an inference, then this inference will actually be reflected in a domain change.

A propagator is *domain complete* (or simply *complete*) for a constraint c if it establishes domain consistency. More formally, $\mathrm{dom}(c)$ is the strongest domain including all valid assignments of a constraint, defined as $\min\{d \in \mathit{Dom} \mid c \subseteq d\}$. The minimum exists as domains are closed under intersection, and the definition is non-trivial because not every constraint can be captured by a domain. Now, for a constraint c and a domain d, $\mathrm{dom}(c \cap d)$ refers to removing all values from d not supported by the constraint c. A propagator p is complete for a constraint c iff for all domains d, we have $p(d) \subseteq \mathrm{dom}(c \cap d)$. A complete propagator thus removes all assignments from d that are inconsistent with c. One of the main results of this paper is that domain completeness is preserved by views.

Theorem 4. If p is complete for c, then $\widehat{\varphi}(p)$ is complete for $\varphi^{-}(c)$.

A propagator is *bounds complete* for a constraint c, if it only affects domain bounds, or only depends on domain bounds for its inferences. For our purposes, we only distinguish bounds($\mathbb{Z}$) and bounds($\mathbb{R}$) completeness (see [3] for an overview). Our definitions of bounds completeness are based on the *strongest convex domain* that contains a constraint, $\mathrm{conv}(c) = \min\{d \in \mathit{Dom} \mid c \subseteq d \text{ and } d \text{ convex}\}$. A convex domain maps each variable to an interval, so that $\mathrm{conv}(c)(x) = \{\min_{a \in c}(a(x)), \ldots, \max_{a \in c}(a(x))\}$. Following Benhamou [1] and Maher [5], we define that p is bounds($\mathbb{Z}$) complete for c iff $p(d) \subseteq \mathrm{conv}(c \cap \mathrm{conv}(d))$, and p is bounds($\mathbb{R}$) complete for c iff $p(d) \subseteq \mathrm{conv}(c_{\mathbb{R}} \cap \mathrm{conv}_{\mathbb{R}}(d))$, where $\mathrm{conv}_{\mathbb{R}}(d)$ is the convex hull of d in $\mathbb{R}$, and $c_{\mathbb{R}}$ is c relaxed to $\mathbb{R}$.

The result for domain completeness does not carry over directly to bounds completeness: we can only derive bounds complete propagators using views that satisfy certain additional properties. A constraint c is a φ *constraint* iff for all $a \in c$, there is a $b \in Asn$ such that $a = \varphi_{Asn}(b)$. A view φ is *interval injective* iff $\varphi^{-}(\text{conv}(c)) = \text{conv}(\varphi^{-}(c))$ for all φ constraints c. It is *interval bijective* iff it is interval injective and $\varphi(\text{conv}(d)) = \text{conv}(\varphi(d))$ for all domains d. The following table summarizes how completeness depends on view bijectivity:

propagator	*interval bijective view*	*interval injective view*	*arbitrary view*
domain	domain	domain	domain
bounds($\mathbb{Z}$)	bounds($\mathbb{Z}$)	bounds($\mathbb{R}$)	weakly
bounds($\mathbb{R}$)	bounds($\mathbb{R}$)	bounds($\mathbb{R}$)	weakly

Minus and offset views are interval bijective. A scale view for $a \in \mathbb{Z}$ on x is always interval injective and only interval bijective if $a = 1$ or $a = -1$. An important consequence is that a bounds($\mathbb{Z}$) complete propagator for the constraint $\sum_i x_i = c$, when instantiated with scale views for the x_i, results in a bounds($\mathbb{R}$) complete propagator for $\sum_i a_i x_i = c$.

Views are related to *indexicals* [2, 10], propagators that prune a single variable and are defined over range expressions. However, views are not used to define propagators, but to derive new propagators from existing ones. Allowing the full expressivity of indexicals for views would imply giving up our completeness results. Another related concept are arithmetic expressions (as found in ILOG Solver [6]). In contrast to views, expressions are used for modeling, not for propagation, and, like indexicals, yield no completeness guarantees.

4 Extended Properties of Derived Propagators

A derived propagator permits further derivation: $\widehat{\varphi}(\widehat{\varphi'}(p))$ is perfectly acceptable, properties like correctness and completeness carry over. For instance, we can derive a propagator for $x - y = c$ from a propagator for $x + y = 0$ by combining an offset and a minus view on y.

A propagator is idempotent iff $p(p(d)) = p(d)$ for all domains d. Some systems require all propagators to be idempotent, others apply optimizations if the idempotence of a propagator is known [7].

Theorem 5. *If a propagator is derived from an idempotent propagator, the result is idempotent again: If $p(p(d)) = p(d)$ for a propagator p and a domain d, then, for any view φ, $\widehat{\varphi}(p)(\widehat{\varphi}(p)(d)) = \widehat{\varphi}(p)(d)$.*

A propagator is subsumed for a domain d iff for all stronger domains $d' \subseteq d$, $p(d') = d'$. Subsumed propagators do not contribute any propagation in the remaining subtree of the search, and can therefore be removed. Deciding subsumption is coNP-complete in general, but for most propagators an approximation can be decided easily. This can be used to optimize propagation.

Theorem 6. *p is subsumed by $\varphi(d)$ iff $\widehat{\varphi}(p)$ is subsumed by d.*

An alternative model of views is to regard a view φ as additional *view constraints*, implementing the *decomposition* of a constraint.

Example. Assume the equality constraint $c \equiv (x = y)$. In order to propagate $c' \equiv (x = y + 1)$, we could use a domain complete propagator p for c and a view φ with $\varphi_x(v) = v$, $\varphi_y(v) = v+1$. The alternative model would contain additional variables x' and y', a view constraint $c_{\varphi,x}$ for $x' = x$, a view constraint $c_{\varphi,y}$ for $y' = y + 1$, and $c[x/x', y/y']$, which enforces equality of x' and y'.

In general, every view constraint $c_{\varphi,i}$ shares exactly one variable with c and no variable with any other $c_{\varphi,i}$. Thus, the constraint graph is Berge-acyclic, and we can reach a fixpoint by first propagating all the $c_{\varphi,i}$, then propagating $c[x_1/x'_1, \ldots, x_n/x'_n]$, and then again propagating the $c_{\varphi,i}$. This is exactly what $\varphi^- \circ p \circ \varphi$ does. In this sense, views can be seen as a way to specify a *perfect order of propagation*, which is usually not possible in constraint programming systems. Furthermore, if $\widehat{\varphi}(p)$ is domain complete for $\varphi^-(c)$, it achieves *path consistency* for $c[x_1/x'_1, \ldots, x_n/x'_n]$ and all the $c_{\varphi,i}$ in the decomposition model.

Acknowledgements. We thank Mikael Lagerkvist and Gert Smolka for fruitful discussions about views and helpful comments on a draft of this paper.

References

1. F. Benhamou. Heterogeneous Constraint Solving. In *Proceedings of the fifth International Conference on Algebraic and Logic Programming (ALP'96)*, volume 1139 of *LNCS*, pages 62–76. Springer, 1996.
2. M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. H. Hartel, and H. Kuchen, editors, *Programming Languages: Implementations, Logics, and Programs, 9th International Symposium, PLILP'97*, volume 1292 of *LNCS*, pages 191–206, Southampton, UK, 1997. Springer.
3. C. W. Choi, W. Harvey, J. H. M. Lee, and P. J. Stuckey. Finite domain bounds consistency revisited. In A. Sattar and B.-H. Kang, editors, *AI 2006: Advances in Artificial Intelligence*, volume 4304 of *LNCS*, pages 49–58. Springer, 2006.
4. Gecode: Generic constraint development environment, 2008. Available as an open-source library from `http://www.gecode.org`.
5. M. J. Maher. Propagation completeness of reactive constraints. In *ICLP '02: Proceedings of the 18th International Conference on Logic Programming*, volume 2401 of *LNCS*, pages 148–162, London, UK, 2002. Springer.
6. J.-F. Puget and M. Leconte. Beyond the glass box: Constraints as objects. In J. Lloyd, editor, *Proceedings of the International Symposium on Logic Programming*, pages 513–527, Portland, OR, USA, Dec. 1995. The MIT Press.
7. C. Schulte and P. J. Stuckey. Efficient constraint propagation engines. *Transactions on Programming Languages and Systems*, 2008. To appear.
8. C. Schulte and G. Tack. Views and iterators for generic constraint implementations. In *Recent Advances in Constraints (2005)*, volume 3978 of *LNAI*, pages 118–132. Springer, 2006.
9. C. Schulte and G. Tack. Perfect derived propagators, June 2008. Available from `http://arxiv.org/abs/0806.1806`.
10. P. Van Hentenryck, V. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(FD). *The Journal of Logic Programming*, 37(1–3):139–164, Oct. 1998.