ID2204: Constraint Programming

# Constraint Propagation

Lecture 04, 2010-04-01

Christian Schulte
cschulte@kth.se

Software and Computer Systems
School of Information and Communication Technology
KTH – Royal Institute of Technology
Stockholm, Sweden

KTH Information and Communication Technology

# Naïve Constraint Propagation

# Naïve Constraint Propagation

- Looking for

  propagate : $M \times S \rightarrow S$

  performing constraint propagation
  - start from some initial store
  - return store on which all propagation has been performed
  - ignore efficiency, focus on principle idea

M = set of all models
S = set of all stores

# Naïve Propagation Function

V = set of decision variables
U = universe = common domain
P = set of propagators of model
s = store to start from

```
propagate((V,U,P), s)
    while p∈P and p(s) ≠ s do
        s := p(s);
    return s;
```

- What is returned as result?
- Does it terminate?

# Termination

- Consider store $s_i$ at $i$-th iteration of loop with $s_o$ initial store

  $$s_{i+1} < s_i$$

- That is, $s_i$ form strictly decreasing sequence: cannot be infinite
  - remember: (S,<) is well-founded!
- Loop terminates!

# Result Computed

- Assume `propagate((`$V$`,`$U$`,`$P$`),`$s$`)=`$s'$

$$\text{sol}((V,U,P),s) = \text{sol}((V,U,P),s')$$

no solutions removed

$$\text{for all } p \in P\text{: } p(s') = s'$$

no further propagation possible

the weakest simultaneous fixpoint: unique (by mono.)

NB: a strongest simultaneous fixpoint would be a solution (hence not unique), which would violate solution preservation

# Weakest Simultaneous Fixpoint

- Assume `propagate`$((V,U,P),s) = s'$

  Then

  $s'$ weakest sim. fixpoint with $s' \leq s$

  that is

  for all $p \in P$ $\quad p(s') = s'$

  - clear, follows from termination of loop

  weakest fixpoint?

  - any other fixpoint is stronger

# Why Naïve?

- Always searches all propagators of model for propagator which can contract strictly
  - maintain propagators which are known to have fixpoint computed
  - might have to find out by having propagators which do no contraction
  - take variables into account which connect two propagators

# Realistic Constraint Propagation

# Improving Propagation

- Idea: propagator narrows domain of some (few) variables
  - re-propagate only propagators sharing modified variables
- Maintain a set of “dirty” propagators
  - not known whether at fixpoint for current store
  - all other propagators have fixpoint computed
  - only propagate "dirty" propagators

# Propagator Variables

- Variables var(*p*) of propagator *p*
  - variables of interest
- No input considered from other variables
- No output computed on other variables

# Variable Dependencies

- No output on other variables

  for all $s \in S$, for all $x \in (V\text{-var}(p))$

  $$p(s)(x)=s(x)$$

- No input from other variables

  for all $s_1, s_2 \in S$

  if (for all $x \in \text{var}(p)$: $s_1(x)=s_2(x)$),

  then (for all $x \in \text{var}(p)$:

  $$p(s_1)(x)=p(s_2)(x))$$

# Idea: Improved Propagation

- maintain set *N* of "dirty" propagators
- choose propagator to run next from *N* and remove from *N*
- compute modified variables
- add propagators sharing variables with modified variables to *N*
  - including the running propagator!
    (as propagators need not be idempotent)

# Improved Propagation

propagate$((V,U,P), s_o)$

$s := s_0$; $N := P$;

**while** $N \neq \varnothing$ **do**

choose $p \in N$;

$s' := p(s)$; $N := N - \{p\}$;

$MV := \{\, x \in V \mid s(x) \neq s'(x) \,\}$; (modified variables)

(dependent propagators) $DP := \{\, q \in P \mid$ exists $x \in \text{var}(q)$: $x \in MV \,\}$;

$N := N \cup DP$; (maintain set N of dirty propagators)

$s := s'$;

**return** $s$;

# Questions

- What does it compute
  - does it compute simultaneous fixpoint?
  - the weakest?
  - important: loop invariant
- Termination?
  - stores are not any longer strictly stronger

# Summary:

- Propagators
  - compute with stores
  - are contracting and monotonic
  - maintain solutions
  - strong enough to decide for assignments
- Naïve propagation
  - terminates
  - computes weakest simultaneous fixpoint
- Dependency directed propagation
  - if variables shrink, rerun dependent propagators only
  - computes weakest simultaneous fixpoint
  - terminates

domains of some

# Improving Propagation Further

# General Idea

- Essential: knowledge on fixpoint for a propagator
- So far: only implicit knowledge
- Here: let us make knowledge explicit
  - propagators provide information

# We Are Done! What Now?

- Suppose the following propagator

  $$p(s) = \{x \rightarrow (s(x) \cap \{1,2,3\})\}$$

  - implements domain constraint $x \in \{1,2,3\}$
- After executing $p$ once, no further execution needed:

  $$\text{if } s' \leq p(s) \text{ then } p(s')=s'$$

- We can safely delete $p$ from model
  - otherwise, pointless re-execution!

# Subsumed Propagators

Definition:

- Propagator $p$ ***subsumed*** by store $s$, iff

  for all $s' \leq s : p(s')=s'$

  - all stronger stores are fixpoints
  - $p$ entailed by $s$
  - $s$ subsumes $p$ ($s$ entails $p$)

# Reminder: Propagator for $\leq$

- Propagator $p_{\leq}$ for $x \leq y$

  $p_{\leq}(s) =$

  $\{ x \rightarrow \{ n \in s(x) \mid n \leq \max(s(y)) \},$

  $y \rightarrow \{ n \in s(y) \mid n \geq \min(s(x)) \}\}$

Example:

p_<= is subsumed by {x : {1,2,3}, y : {3,4,5}}.

p_<= is not subsumed by {x : {1,2,3,4}, y : {3,4,5}}.

# We Are Done! What Next?

- After executing $p_{\leq}$ on store s we have

  $$p_{\leq}(p_{\leq}(s)) = p_{\leq}(s)$$

  - max($s$($y$)) does not change!
  - min($s$($x$)) does not change!
- What happens: as var($p_{\leq}$)={$x$,$y$}, $p_{\leq}$ is added to *DP* and hence to N
  - but: $s'$ is fixpoint for $p_{\leq}$
  - no need to include in *DP*

# First Attempt: Idempotent Functions

Definition:

- A function $f \in X \rightarrow X$ is ***idempotent***, if

  for all $x \in X$:   $f(f(x)) = f(x)$

- Very strong property for a propagator: required for all stores!

An example of a non-idempotent propagator is given in Example 2.9 on page 19 of Course Notes.

A domain consistent propagator is necessarily idempotent.

A bounds(Z) consistent propagator is not necessarily idempotent.

# Second Attempt: Weak Idempotence

- A function $f \in X \rightarrow X$ is ***idempotent on*** $x \in X$ if x is a fixpoint of f:

  $$f(f(x)) = f(x)$$

  - statement on just one element

- For a propagator: if $p$ is idempotent on $s$, it does not mean that $p$ is idempotent on $s'$ with $s' \leq s$

# How to Find Out?

- Given store *s* and propagator *p*
- Does *s* subsume *p*?
  - try all *s*' < *s*: way tooo costly
- Is *p* idempotent on *s*?
  - apply *p* to *s*: that is what we tried to avoid in 1st place

# Status Messages

- Solution: propagator returns status and tells result

  propagator $p$ is function

  $$p \in S \rightarrow SM \times S$$

  with

  $$SM := \{\text{fix}, \text{nofix}, \text{subsumed}\}$$

# Propagator with Status

- Assume propagator *p* and store *s*

  if $p(s)$ = (fix, $s'$), then

  $s'$ is fixpoint for *p*

  if $p(s)$ = (subsumed, $s'$), then

  $s'$ subsumes *p*

  if $p(s)$ = (nofix, $s'$), then

  no further knowledge

  always safe (as before)

# Propagator for ≤ with Subsumption

- Propagator $p_\leq$ for $x \leq y$

  $p_\leq(s) =$

  **if** $\max(s(x)) \leq \min(s(y))$ **then**

  (subsumed, s)

  **else**

  (fix,

  $\{ x \rightarrow \{ n \in s(x) \mid n \leq \max(s(y)) \},$

  $y \rightarrow \{ n \in s(y) \mid n \geq \min(s(x)) \}\})$

but subsumption could also be tested for in the else case!

# Propagator for ≤ with Subsumption (better version)

- Propagator $p_{\leq}$ for $x \leq y$

$ep_{\leq}(s)$ = **let** $s'$ = $p_{\leq}(s)$ **in**

**if** $\max(s'(x)) \leq \min(s'(y))$ **then**

(subsumed, $s'$)

**else**

(fix~~pt~~, $s'$)

# What to Return?

- Propagation function now also needs to return the set of propagators
    - in case of subsumption, propagators are removed

# Improved Propagation

```
propagate((V,U,P), s0)
    s := s0; N := P;
    while N ≠ ∅ do
        choose p ∈ N;
        (sm,s'):= p(s); N := N – {p};
        if sm=subsumed then P := P – {p}; end
        MV := { x ∈ V | s(x) ≠ s'(x) };
        DP := { q ∈ P | exists x ∈ var(q): x ∈ MV };
        if sm=fix then DP := DP – {p}; end
        N := N ∪ DP;
        s := s';
    return (P,s);
```

# Correctness

- Are the optimizations correct?
- How to prove:
  - invariant is still invariant
  - solutions remain the same
  - still computes the same

argument: fixpoints!

# Propagation Events

# Propagation Events

- Many propagators
  - simple to decide whether still at fixpoint for changed domain
  - based on ***how*** domain has changed
- How domain changes described by

  ***propagation event***

  or just event

# Propagator for $\leq$

- Propagator $p_{\leq}$ for $x \leq y$

  $p_{\leq}(s) =$

  $\{\, x \rightarrow \{\, n \in s(x) \mid n \leq \max(s(y)) \,\},$

  $y \rightarrow \{\, n \in s(y) \mid n \geq \min(s(x)) \,\}\}$

  - must be propagated only if $\max(s(y))$ or $\min(s(x))$ changes

# Propagator for ≠

- Propagator $p_{\neq}$ for $x \neq y$

  $p_{\neq}(s) =$

  $\{ x \rightarrow s(x) - \text{single}(s(y)),$

  $y \rightarrow s(y) - \text{single}(s(x))\}$

  - where: $\text{single}(\{n\}) = \{n\}$

    $\text{single}(N) = \emptyset$ (otherwise)

  - must be propagated only if $x$ or $y$ become assigned

# Events

- Typical events
  - fix(*x*) *x* becomes assigned
  - min(*x*) minimum of *x* changes
  - max(*x*) maximum of *x* changes
  - any(*x*) domain of *x* changes
- Clearly overlap
  - fix(*x*) occurs: min(*x*) or max(*x*) occur
    any(*x*) occurs
  - min(*x*) or max(*x*) occur: any(*x*) occurs

Do not mix up the "fix" status message
with the "fix(x)" propagation event.

# Events on Store Change

events($s$,$s'$) =

$$\{\ \text{any}(x)\ \mid s'(x) \subset s(x)\ \} \cup$$
$$\{\ \text{min}(x)\ \mid \text{min}\ s'(x) > \text{min}\ s(x)\ \} \cup$$
$$\{\ \text{max}(x) \mid \text{max}\ s'(x) < \text{max}\ s(x)\ \} \cup$$
$$\{\ \text{fix}(x)\ \mid |s'(x)|=1 \text{ and } |s(x)|>1\ \}$$

- where $s' \leq s$

# Events: Example

- Given stores
  - $s = \{\ x_1 \to \{1,2,3\},\quad x_2 \to \{3,4,5,6\},$
    $x_3 \to \{0,1\},\quad x_4 \to \{7,8,10\}\}$
  - $s' = \{\ x_1 \to \{1,2\},\quad x_2 \to \{3,5,6\},$
    $x_3 \to \{1\},\quad x_4 \to \{7,8,10\}\}$
- Then events($s$,$s'$) =

  { max($x_1$), any($x_1$),
  any($x_2$),
  fix($x_3$), min($x_3$), any($x_3$)}

# Events are Monotonic

- If $s'' \le s'$ and $s' \le s$ then

  $$\text{events}(s,s'') = \text{events}(s,s') \cup \text{events}(s',s'')$$

- Event occurs on change from $s$ to $s''$
  - occurs on change from $s$ to $s'$, or
  - occurs on change from $s'$ to $s''$

# Event Sets: First Requirement

- Event set for propagator $p$: es($p$)
  - for all stores $s'$ and $s$ with $s' \leq s$ and $s'(x) = s(x)$ for all $x \in V\text{-var}(p)$

    if $p(s)=s$ and $p(s')\neq s'$ then

    es($p$) $\cap$ events($s$,$s'$) $\neq \emptyset$
  - if store $s$ is fixpoint and changes to non-fixpoint $s'$, then events from $s$ to $s'$ must be included in es($p$)

# Event Sets: Second Requirement

- Event set for propagator $p$: es($p$)
  - for all stores $s$ with $p(p(s)) \neq p(s)$:

    $$\text{es}(p) \cap \text{events}(s, p(s)) \neq \emptyset$$
  - if propagator does not compute fixpoint on store $s$, then events from $s$ to $p(s)$ must be included in es($p$)
  - does not occur for idempotent propagators (which compute fixpoints in one go)

# Propagator for ≤

- Propagator $p_{\leq}$ for $x \leq y$

  $p_{\leq}(s) =$

  $\{\ x \rightarrow \{\ n \in s(x) \mid n \leq \max(s(y))\ \},$

  $y \rightarrow \{\ n \in s(y) \mid n \geq \min(s(x))\ \}\}$

  - good one: $\text{es}(p_{\leq}) = \{\ \max(y), \min(x)\ \}$
  - but also: $\text{es}(p_{\leq}) = \{\ \text{any}(y), \text{any}(x)\}$

# Propagator for ≠

- Propagator $p_{\neq}$ for $x \neq y$

  $p_{\neq}(s) =$

  $\{ x \rightarrow s(x) - \text{single}(s(y)),$

  $y \rightarrow s(y) - \text{single}(s(x))\}$

  - where: single($\{n\}$) $= \{n\}$
    single($N$) $= \emptyset$ (otherwise)

  - good one: es($p_{\neq}$) = { fix($y$), fix($x$) }
  - but also: es($p_{\neq}$) = { any($y$), any($x$)}

# Taking Advantage from Event Sets

- Base decision of propagators to re-propagate on event sets rather than on modified variables

$$DP := \{\ q \in P \mid \text{events}(s,s') \cap \text{es}(q) \neq \emptyset \};$$

Note that the MV set is not needed any more.

# More Optimizations

# Priorities

- Choose propagator
  - according to cost: cheapest first
  - according to expected impact
  - general: first-in first-out (queue)

# Propagator Rewriting

- Another observation: propagator for

  $\max(x,y)=z$

  and values for $x$ are smaller than for $y$
- Replace by propagator for $y=z$

# Summary: Optimizing Propagation

- Fixpoint knowledge avoids useless execution
  - idempotence, subsumption, events
  - knowledge provided by propagator

- More details on optimizing propagation and propagation in systems

  **Finite Domain Constraint Programming Systems,** Christian Schulte, Mats Carlsson.

  **In:** Francesca Rossi, Peter van Beek, Toby Walsh, editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, pages 495-526. Elsevier Science Publishers, 2006.