Principles of Constraint Programming

Krzysztof R. Apt

Chapter 4
Some Complete Constraint Solvers

Objectives

- Introduce a simple proof theoretic framework.
- Use it to define complete solvers.
- Discuss **Martelli-Montanari** unification algorithm for solving term equations.
- Discuss GAUSS–JORDAN ELIMINATION and GAUSSIAN ELIMINATION algorithms for solving linear equations over reals.

Proof Theoretic Framework

- **Rules** that transform CSP's

$$\frac{\langle \mathcal{C} \ ; \ \mathcal{DE} \rangle}{\langle \mathcal{C}' \ ; \ \mathcal{DE}' \rangle}$$

- A rule

$$\frac{\phi}{\psi}$$

  is **equivalence preserving** if ϕ and ψ are equivalent.

- All considered rules will be equivalence preserving.

Types of Rules

Domain reduction rules

- $\mathcal{DE} := x_1 \in D_1, \ldots, x_n \in D_n$,
- $\mathcal{DE}' := x_1 \in D'_1, \ldots, x_n \in D'_n$,
- for $i \in [1..n]$
$$D'_i \subseteq D_i,$$
- $\mathcal{C}'$: restriction of all constraints in $\mathcal{C}$ to the domains $D'_1, \ldots, D'_n$.

Transformation rules

- Not domain reduction rules,
- $\mathcal{C}' \neq \emptyset$,
- $\mathcal{DE}'$ extends $\mathcal{DE}$.

Examples: Domain reduction rules

- *LINEAR DISEQUALITY*

$$\frac{\langle x < y \ ; \ x \in [l_x..h_x], y \in [l_y..h_y] \rangle}{\langle x < y \ ; \ x \in [l_x..h'_x], y \in [l'_y..h_y] \rangle}$$

where

$$h'_x = min(h_x, h_y - 1),$$
$$l'_y = max(l_y, l_x + 1.$$

- *EQUALITY*

$$\frac{\langle x = y \ ; \ x \in D_x, y \in D_y \rangle}{\langle x = y \ ; \ x \in D_x \cap D_y, y \in D_x \cap D_y \rangle}$$

- *DISEQUALITY*

$$\frac{\langle x \neq y \ ; \ x \in D, y = a \rangle}{\langle \ ; \ x \in D - \{a\}, y = a \rangle}$$

(domain expression $y = a$ stands for $y \in \{a\}$.)

Examples: Transformation rules

- *DISEQUALITY TRANSFORMATION*

$$\frac{\langle s \neq t \ ; \ \mathcal{DE} \rangle}{\langle x \neq t, \ x = s \ ; \ \mathcal{DE}, x \in \mathcal{Z} \rangle}$$

where

– s is not a variable,

– $\mathcal{DE}$ includes all variables present in s and t,

– x does not appear in $\mathcal{DE}$.

- *VARIABLE ELIMINATION*

$$\frac{\langle \mathcal{C} \ ; \ \mathcal{DE}, x = a \rangle}{\langle \mathcal{C}\{x/\overline{a}\} \ ; \ \mathcal{DE}, x = a \rangle}$$

where x occurs in $\mathcal{C}$.

$\mathcal{C}\{x/\overline{a}\}$: constraints obtained from $\mathcal{C}$ by substituting each occurrence of x by $\overline{a}$.

An instance:

$$\frac{\langle 3xy^2 + 5xy - 5yz \leq 6 \ ; \ x \in [0..100], y = 2, z \in [0..100] \rangle}{\langle 22x - 10z \leq 6 \ ; \ x \in [0..100], y = 2, z \in [0..100] \rangle}$$

Rule Applications

- **Application** of a rule (informally): replace in a CSP the part that matches the premise by the conclusion.
- **Relevant application** of a rule (informally):

  the result **differs** from the initial CSP.
- A CSP $\mathcal{P}$ is **closed under the applications of R** if

  – R cannot be applied to $\mathcal{P}$

  or

  – no application of it to $\mathcal{P}$ is relevant.

Recap: Solved and Failed CSP's

- A constraint is **solved** if it equals the Cartesian product of the domains of its variables.
- CSP is **solved** if all its constraints are solved.
- CSP is **failed** if

  either

  it contains the false constraint $\perp$

  or

  some of its domains or constraints is empty.

Derivations

Given: a finite set of proof rules.

- **Derivation**: a sequence of CSP's s.t. each is obtained from the previous one by an application of a proof rule.
- A finite derivation is called
  - **successful**: last element is a first solved CSP in this derivation,
  - **failed**: last element is a first failed CSP in this derivation,
  - **stabilising**: last element is a first CSP closed under the applications of the proof rules.

Derivation: Example

Take

- *EQUALITY*

$$\frac{\langle x = y \; ; \; x \in D_x, y \in D_y \rangle}{\langle x = y \; ; \; x \in D_x \cap D_y, y \in D_x \cap D_y \rangle}$$

- *DISEQUALITY*

$$\frac{\langle x \neq y \; ; \; x \in D, y = a \rangle}{\langle \; ; \; x \in D - \{a\}, y = a \rangle}$$

and consider CSP

$$\langle x = y, y \neq z, z \neq u; \\ x \in \{a, b, c\}, y \in \{a, b, d\}, z \in \{a, b\}, u = b \rangle.$$

Derivation: Example, ctd

$$\langle x = y, y \neq z, z \neq u;$$
$$x \in \{a, b, c\}, y \in \{a, b, d\}, z \in \{a, b\}, u = b\rangle.$$

Apply *EQUALITY* rule:

$$\langle x = y, y \neq z, z \neq u \ ; \ x \in \{a, b\}, y \in \{a, b\}, z \in \{a, b\}, u = b\rangle.$$

Apply *DISEQUALITY* rule to $z \neq u$

$$\langle x = y, y \neq z, z \neq u \ ; \ x \in \{a, b\}, y \in \{a, b\}, z = a, u = b\rangle.$$

Apply *DISEQUALITY* rule to $y \neq z$

$$\langle x = y, y \neq z, z \neq u \ ; \ x \in \{a, b\}, y = b, z = a, u = b\rangle.$$

Apply *EQUALITY* rule

$$\langle x = y, y \neq z, z \neq u \ ; \ x = b, y = b, z = a, u = b\rangle.$$

Last CSP is **solved**:
the derivation is **successful**.

Term Equations

Alphabet

It consists of

- **variables**,
- **function symbols**, each with a fixed **arity**,
- **parentheses**: "(" and ")",
- **comma**, that is: "," .

Terms

Defined inductively as follows.

- a variable is a **term**,
- if f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a **term**.

Note: Every constant is a term.

Substitutions

- Finite mappings from **variables** to **terms**.
  To each variable x in its domain a term **different** from x is assigned.
- Written as

$$\{x_1/t_1, \ldots, x_n/t_n\}$$

where

– $x_1, \ldots, x_n$ are different variables,
– $t_1, \ldots, t_n$ are terms,
– for $i \in [1, n]$, $x_i \not\equiv t_i$.

Applying Substitutions

Given: term s, substitution θ.

$s\theta$: result of **applying θ to s**.

Replace simultaneously each variable in s by corresponding term from θ.

Example Take language of arithmetic expressions in prefix form.

$s := +(\cdot(x, 7), \cdot(4, y)),$

$\theta := \{x/0, y/+(z, 2)\}$

Then

$$s\theta = +(\cdot(0, 7), \cdot(4, +(z, 2))).$$

Composing Substitutions

- **Given**: substitutions θ and η.

  $\theta\eta$: **composition of θ and η.**

  $$(\theta\eta)(x) := (x\theta)\eta.$$

  Note: $\theta\eta$ is a substitution.

  Example Take

  $\theta := \{u/z, x/3, y/f(x,1)\}$,

  $\eta := \{x/4, z/u\}$.

  Then

  $$\theta\eta = \{x/3, y/f(4,1), z/u\}.$$

- θ is **more general than** τ if for some substitution η

  $$\tau = \theta\eta.$$

  Example Take

  $\theta := \{y/g(x,a), z/b\}$,

  $\tau := \{x/c, y/g(c,a), z/b\}$.

  θ is more general than τ since

  $\{x/c, y/g(c,a), z/b\} = \{y/g(x,a), z/b\}\{x/c\}$.

Unification

- θ is a **unifier** of s and t if

$$s\theta \equiv t\theta.$$

- θ is a **most general unifier** (**mgu**) of s and t if
  - θ is a unifier of s and t,
  - θ is more general than all unifiers of s and t.

Example

Take $f(g(x, a), z)$ and $f(y, b)$. Then

- $\{x/c, y/g(c, a), z/b\}$ is one of their unifiers.
- $\{y/g(x, a), z/b\}$ is an mgu of $f(g(x, a), z)$ and $f(y, b)$.

Sets of Term Equations

- θ is a **unifier** of a set of term equations $\{s_1 = t_1, \ldots, s_n = t_n\}$ if θ is a unifier of s_i and t_i for $i \in [1..n]$.
- θ is an **mgu** of E if
  - θ is a unifier of E,
  - θ is more general than all unifiers of E.
- Two sets of equations are **equivalent** if they have the same set of unifiers.

Connection with CSP's

- **Variable domains**: $\mathcal{T}$, the set of all terms in the considered alphabet.
- $s = t$ with variables $x_1, \ldots, x_n$ represents the constraint

$$\{(x_1\eta, \ldots, x_n\eta) \mid \eta \text{ a unifier of } s \text{ and } t\}.$$

- $\{s_1 = t_1, \ldots, s_k = t_k\}$ with variables $x_1, \ldots, x_n$ represents

$$\langle s_1 = t_1, \ldots, s_k = t_k \,;\, x_1 \in \mathcal{T}, \ldots, x_n \in \mathcal{T}\rangle.$$

Note E: finite set of term equations with the variables $x_1, \ldots, x_n$. Then

$$Sol(\langle E \,;\, x_1 \in \mathcal{T}, \ldots, x_n \in \mathcal{T}\rangle) =$$
$$\{(x_1\eta, \ldots, x_n\eta) \mid \eta \text{ a unifier of } E\}.$$

UNIF Proof System

DECOMPOSITION

$$\frac{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)}{s_1 = t_1, \ldots, s_n = t_n}$$

FAILURE 1

$$\frac{f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)}{\perp}$$

where $f \not\equiv g$,

DELETION

$$\frac{x = x}{}$$

UNIF Proof System, Ctd

$$TRANSPOSITION$$

$$\frac{t = x}{x = t}$$

where t is not a variable,

$$SUBSTITUTION$$

$$\frac{x = t,\ E}{x = t,\ E\{x/t\}}$$

where $x \notin Var(t)$ and $x \in var(E)$,

$$FAILURE\ 2$$

$$\frac{x = t}{\perp}$$

where $x \in Var(t)$ and $x \not\equiv t$.

Lemma Each rule of *UNIF* is **equivalence preserving** (w.r.t. sequence of the variables present in the rule premise).

A Derivation in *UNIF*

Selected equations underlined.

Take

$E := \{k(z, f(x, b, z)) = k(h(x), f(g(a), y, z))\}$.

Using *DECOMPOSITION* rule we get

$$\{z = h(x), \underline{f(x, b, z) = f(g(a), y, z)}\}.$$

Using *DECOMPOSITION* rule again we get

$$\{z = h(x), x = g(a), \underline{b = y}, z = z\}.$$

Using *TRANSPOSITION* rule we get

$$\{z = h(x), x = g(a), y = b, \underline{z = z}\}.$$

Using *DELETION* rule we get

$$\{z = h(x), \underline{x = g(a)}, y = b\}.$$

Using *SUBSTITUTION* rule we get

$$\{z = h(g(a)), x = g(a), y = b\}.$$

No rule applies at this stage.

$\{z/h(g(a)), x/g(a), y/b\}$ is an **mgu** of E.

Martelli-Montanari Algorithm

Given:

– CSP $\mathcal{P} := \langle \mathcal{C} \; ; \; \mathcal{DE} \rangle$

– rule

$$\mathcal{R} := \frac{\langle \mathcal{C} \; ; \; \mathcal{DE} \rangle}{\langle \mathcal{C}' \; ; \; \mathcal{DE}' \rangle}$$

– $\langle \mathcal{C}' \; ; \; \mathcal{DE}' \rangle$ is the result of applying $\mathcal{R}$ to $\mathcal{P}$.

– This rule application of $\mathcal{R}$ is called **global**.

Martelli-Montanari Algorithm

- *UNIF* proof rules.
- All applications of the *SUBSTITUTION* rule global.

Correctness

Theorem

Given: finite set of term equations E.

- Martelli–Montanari algorithm always terminates.
- If E has a unifier, then each execution of the algorithm terminates with a set of equations that determines an mgu of E.

  Otherwise each execution terminates with a set containing the false constraint $\perp$.

Termination: Proof Sketch

Consider the lexicographic ordering $\prec_3$ on N^3.

$$(m_1, m_2, m_3) \prec_3 (n_1, n_2, n_3) \text{iff}$$

$$\begin{array}{ll} & m_1 < n_1 \\ \text{or} & m_1 = n_1 \wedge m_2 < n_2 \\ \text{or} & m_1 = n_1 \wedge m_2 = n_2 \wedge m_3 < n_3. \end{array}$$

- A variable x **solved in E** if for some term t, $x = t \in E$ and this is the only occurrence of x in E.
- A variable **unsolved** if it is not solved.

$uns(E)$	–	the # of unsolved variables in E,
$lfun(E)$	–	total # of occurrences of fun. syms on the LHS of an equation in E,
$card(E)$	–	the # of equations in E.

– Each rule application reduces the triple

$$(uns(E), lfun(E), card(E))$$

in the lexicographic ordering $\prec_3$.

– For the *SUBSTITUTION* rule it holds only for global applications.

Linear Equations over Reals

Alphabet

- each real number is a constant,
- for each real number r unary function symbol '$r\cdot$',
- binary function symbol '$+$', (written in the infix notation).

Linear expressions and equations

- **Linear expression over reals**: a term in this alphabet.
- **Linear equation over reals**:

$$s = t,$$

  s, t linear expressions.

Normal Forms

Assume ordering $\prec$ on the variables.

- Linear expression in **normal form**:

  $$\Sigma_{i=1}^{n} a_i x_i + r,$$

  where $n \geq 0$, $x_1, \ldots, x_n$ are ordered w.r.t. $\prec$.

- Linear equation in **normal form** :

  $$\Sigma_{i=1}^{n} a_i x_i = r,$$

  where $n \geq 0$, $x_1, \ldots, x_n$ are ordered w.r.t. $\prec$.

- Linear equation in **pivot form**:

  $$x = t$$

  if $x \notin Var(t)$ and t is in normal form.

- Each linear equation can be rewritten (**normalises**) to a unique linear equation in normal form.

Substitutions

- **Substitution**: finite mapping from **variables** to linear expressions in **normal form**.
  To each variable x in its domain a linear expression **different** from x is assigned.
- **Application** of a substitution to a linear expression: defined as before.
- **Given**: substitutions θ and γ.
  $\theta\gamma$: **composition of θ and γ**.
  Uniquely determined by
  $$\eta(x) := norm((x\theta)\gamma).$$
- θ is a **unifier** of $s = t$ if $s\theta = t\theta$ normalises to $0 = 0$.
- **mgu**: defined as before.

Pivot Forms

Three types of normal forms:

1. $0 = 0$,
2. $0 = r$ where r is a non-zero real,
3. $\Sigma_{i=1}^{n} a_i x_i = r$, where $n > 0$.

Pivots forms of linear equations

- Each linear equation e normalises to a normal form.
- If it is type 1 or 2, then it has no **pivot form**.
- If it is type 3, then each equation

$$x_j = \Sigma_{i \in [1..j-1] \cup [j+1..n]} - \frac{a_i}{a_j} x_i + \frac{r}{a_j}$$

  is a **pivot form** of e.

LIN Proof System

– $norm(s)$: normal form of s,
– $stand(s = t) \equiv norm(s) = norm(t)$.

$$\mathit{DELETION} \quad \frac{s = v}{}$$

if $s = v$ normalises to $0 = 0$,

$$\mathit{FAILURE} \quad \frac{s = v}{\perp}$$

if $s = v$ normalises to $0 = r$,
r is a non-zero real,

$$\mathit{SUBSTITUTION} \quad \frac{s = v,\ E}{x = t,\ stand(E\{x/t\})}$$

where $x = t$ is a pivot form of $s = v$.

Gauss–Jordan Elimination

Gauss–Jordan Elimination Algorithm

- *LIN* proof rules.
- All applications of the *SUBSTITUTION* rule **global** and condition $x \in Var(E)$ holds.

Theorem

Given: finite set of linear equations E.

- Gauss–Jordan Elimination algorithm always terminates.
- If E has a solution, then each execution of the algorithm terminates with a set of linear equations that determines an mgu of E.

  Otherwise each execution terminates with a set containing the false constraint $\perp$.

Gaussian Elimination

Forward substitution phase:

Repeatedly take the first not yet considered equation from the **left**.

- *DELETION* rule applicable: delete the equation and consider the next equation.
- *FAILURE* applicable: termination with a failure.
- *SUBSTITUTION* rule applicable: apply it taking as E the set of equations lying **to the right** of the current equation.

Backward substitution phase:

Repeatedly take the first not yet considered equation from the **right**.

Apply *SUBSTITUTION* rule taking as E the set of equations lying **to the left** of the current equation.

GAUSSIAN ELIMINATION: Correctness

Theorem

Given: finite set of linear equations E.

- GAUSSIAN ELIMINATION algorithm always terminates.
- If E has a solution, then each execution of the algorithm terminates with a set of linear equations that determines an mgu of E.

  Otherwise each execution terminates with a set containing the false constraint $\perp$.

Objectives

- Introduce a simple proof theoretic framework.
- Use it to define complete solvers.
- Discuss **Martelli-Montanari** unification algorithm for solving term equations.
- Discuss GAUSS–JORDAN ELIMINATION and GAUSSIAN ELIMINATION algorithms for solving linear equations over reals.