Abstract Constraint Programming

Session 5—Abstract Interpretation Workshop

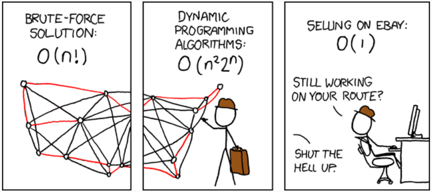
Pierre Talbot
pierre.talbot@uni.lu
20th June 2024
University of Luxembourg

This seminar in a nutshell!

We present the “fusion” of...

Constraint reasoning + Abstract interpretation

(and lattice theory)



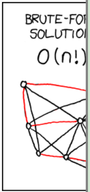
that gives us abstract constraint reasoning.

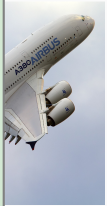
This seminar in a nutshell!

We present the "fusion" of...

etation

ory)





WHY?

- Combining constraint solvers.
- Constructing sound propagators over complex domains.
- Constraint solving on GPUs.

that gives us abstract constraint reasoning.

Background on First-Order Logic

Syntax of First-Order Logic (FOL)

Let $S = \langle X, F, P \rangle$ be a *first-order signature* where X set of variables, F set of function symbols and P set of predicate symbols.

$$
\begin{array}{lll}
\langle Term \rangle ::= & x & \textit{variable } x \in X \\
\mid & f(Term, \ldots, Term) & \textit{function } f \in F
\end{array}
$$

$$
\begin{array}{lll}
\langle \Phi \rangle ::= & p(Term, \ldots, Term) & \textit{predicate } p \in P \\
\mid & \neg \Phi & \textit{negation} \\
\mid & \Phi \diamond \Phi & \textit{connector } \diamond \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\} \\
\mid & \exists x,\ \Phi & \textit{existential quantifier} \\
\mid & \forall x,\ \Phi & \textit{universal quantifier}
\end{array}
$$

- A *theory* is a set of formulas without free variables.
- The substitution $\varphi[x \mapsto t]$ denotes the formula $\varphi \in \Phi$ in which all free occurrences of the variable x in φ have been replaced by the term t.

Semantics of FOL

A *structure* A is a tuple $(\mathbb{U}, [\![]\!]_F, [\![]\!]_P)$ where

1. $\mathbb{U}$ is a non-empty set of elements—called the *universe of discourse*,
2. $[\![]\!]_F$ is a function mapping function symbols $f \in F$ with arity n to interpreted functions $[\![f]\!]_F : \mathbb{U}^n \to \mathbb{U}$, and
3. $[\![]\!]_P$ is a function mapping predicate symbols $p \in P$ with arity n to interpreted predicates $[\![p]\!]_P \subseteq \mathbb{U}^n$.

An assignment is a function $X \to \mathbb{U} \in Asn$ mapping variables to values. Let $\rho \in Asn$, we write $\rho[x \mapsto d]$ the assignment in which we updated the value of x by d in ρ.

Entailment

The syntax and semantics are related by the ternary relation $A \vDash_\rho \varphi$, called the *entailment*, where A is a structure, $\rho \in Asn$ and $\varphi \in \Phi$. It is read as "the formula φ is satisfied by the assignment ρ in the structure A". We first give the interpretation function $[\![]\!]_\rho$ for evaluating the terms of the language:

$$
\begin{aligned}
[\![x]\!]_\rho &= \rho(x) \text{ if } x \in X \\
[\![f(t_1, \ldots, t_n)]\!]_\rho &= [\![f]\!]_F([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho)
\end{aligned}
$$

The relation $\vDash$ is defined inductively as follows:

$$
\begin{array}{ll}
A \vDash_\rho p(t_1, \ldots, t_n) & \text{if } ([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho) \in [\![p]\!]_P \\
A \vDash_\rho \varphi_1 \wedge \varphi_2 & \text{if } A \vDash_\rho \varphi_1 \text{ and } A \vDash_\rho \varphi_2 \\
A \vDash_\rho \varphi_1 \vee \varphi_2 & \text{if } A \vDash_\rho \varphi_1 \text{ or } A \vDash_\rho \varphi_2 \\
A \vDash_\rho \neg\varphi & \text{if } A \vDash_\rho \varphi \text{ does not hold} \\
A \vDash_\rho \exists x,\ \varphi & \text{if there exists } d \in \mathbb{U} \text{ such that } A \vDash_{\rho[x \mapsto d]} \varphi \\
A \vDash_\rho \forall x,\ \varphi & \text{if for all } d \in \mathbb{U}, \text{ we have } A \vDash_{\rho[x \mapsto d]} \varphi
\end{array}
$$

Examples of FOL for Constraint Reasoning

Constraint satisfaction problem (CSP)

CSP $\langle X, D, C \rangle$ is a structured presentation of the logical formula:

$$\bigwedge_{1 \leq i \leq n} x_i \in D_i \wedge \bigwedge_{1 \leq i \leq |C|} C_i$$

Constraint optimization problem (COP)

A COP aims to find the solution of a formula φ maximizing $x \in X$:

$$\varphi \wedge \forall y,\ (\varphi[x \mapsto y] \wedge y \leq x)$$

Multiobjective optimization problem (MOP)

A MOP is a COP with several objectives $x_1, \ldots, x_n \in X$:

$$\varphi \wedge \forall y_1, \ldots, y_n,\ (\varphi[x_1 \mapsto y_1, \ldots, x_n \mapsto y_n] \wedge (x_1 > y_1 \vee \ldots \vee x_n > y_n))$$

Abstract Constraint Reasoning

One Problem, Many Communities, Many Formalisms

Many communities emerged to solve the same problem: find ρ such that $A \models_\rho \varphi$.

BUT they (generally) focus on different fragments of FOL:

- Propositional fragment (SAT): $(a \vee b) \wedge (\neg b \vee c)$ with $a, b, c \in \{0, 1\}$.
- Pseudo-Boolean fragment: $\sum_{1 \leq i \leq n} c_i * a_i \leq c_0$ with $a_i \in \{0, 1\}$ and c_i some integers constants.
- Linear programming (LP): $\sum_{1 \leq i \leq n} c_i * b_i \leq b_0$ with $b_i \in \mathbb{R}$ and c_i some real constants.
- Integer linear programming (ILP): $\sum_{1 \leq i \leq n} c_i * b_i \leq b_0$ with $b_i \in \mathbb{Z}$ and c_i some integer constants.
- Mixed integer linear programming (MILP): $\sum_{1 \leq i \leq n} c_i * b_i \leq b_0$ with $b_i \in \mathbb{Z} \cup \mathbb{R}$ and c_i some integer or real constants.
- Uninterpreted fragment (logic programming).
- Answer set programming.
- Discrete constraint programming: $\langle X, D, C \rangle$ with $D_i \in \mathcal{P}_f(\mathbb{Z})$.
- Continuous constraint programming: $\langle X, D, C \rangle$ with $D_i \in \mathcal{I}(\mathbb{R})$.
- Satisfiability modulo theories (SMT).
- ...

One Theory to Rule Them All?

- SAT [DHK13]
- SMT [DHK14]
- Logic programming [Cou20]
- Constraint programming [Pel+13]
- Linear programming [CH78]
- Multi-objective optimization
- Multilevel programming
- ...

→ Abstract domains

Plan

I. Abstract Constraint Propagation

1. Concrete Domain for First-Order Logic
2. Abstract Propagation

II. Abstract Constraint Search

1. Hoare and Smyth Lattices
2. Abstract Search

III. Conclusion

Concrete Domain for First-Order Logic

Concrete Domain

Definition (Concrete domain)

The concrete domain is the Boolean lattice of assignments $D^\flat = \langle \mathcal{P}(Asn), \subseteq, \cup, \cap, \neg, \{\}, Asn \rangle$ where $\neg$ is the set complement.

Given a structure A, we connect a logical formula to an element of the concrete domain using the interpretation function defined as:

$$\llbracket . \rrbracket^\flat : \Phi \to D^\flat$$
$$\llbracket \varphi \rrbracket^\flat = \{\rho \in Asn \mid A \vDash_\rho \varphi\}$$

A *solution* of the formula φ is an assignment $s \in \llbracket \varphi \rrbracket^\flat$. Applying the interpretation function to a logical formula directly yields the set of all solutions.

Inductive Definition of $[\![.]\!]^\flat$

The Lindenbaum-Tarski algebra is the quotient lattice of quantifier-free first-order formulas defined as $\langle \Phi / \equiv, \leq, \wedge, \vee, \neg, \mathit{true}, \mathit{false} \rangle$ with $[\varphi]_\equiv \leq [\psi]_\equiv$ iff $\psi \vdash \varphi$. We now show that $[\![.]\!]^\flat$ can be constructed inductively.

Theorem

The lattices $\Phi / \equiv$ and D^b are Boolean and $[\![.]\!]^\flat$ is a Boolean homomorphism[1]*. That is, for all formulas φ and ψ, and each predicate p, we have:*

- $[\![\mathit{true}]\!]^\flat = \mathit{Asn}$ and $[\![\mathit{false}]\!]^\flat = \{\}$,
- $[\![p(t_1, \ldots, t_n)]\!]^\flat = \{\rho \in \mathit{Asn} \mid ([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho) \in [\![p]\!]_P\}$,
- $[\![\varphi \wedge \psi]\!]^\flat = [\![\varphi]\!]^\flat \cap [\![\psi]\!]^\flat$,
- $[\![\varphi \vee \psi]\!]^\flat = [\![\varphi]\!]^\flat \cup [\![\psi]\!]^\flat$,
- $[\![\neg\varphi]\!]^\flat = \neg[\![\varphi]\!]^\flat$,
- $\varphi \vdash \psi \Rightarrow [\![\varphi]\!]^\flat \subseteq [\![\psi]\!]^\flat$.

[1] A Boolean homomorphism is a {0,1}-lattice homomorphism between two Boolean lattices.

Closure Operator

The concrete interpretation function $[\![.]\!]^\flat$ can be lifted to a closure operator over the concrete domain defined as follows:

$$\mathcal{F}[\![.]\!] : \Phi \to (D^\flat \to D^\flat)$$
$$\mathcal{F}[\![\varphi]\!]A \triangleq A \cap [\![\varphi]\!]^\flat$$

Closure Operator

The concrete interpretation function $[\![.]\!]^\flat$ can be lifted to a closure operator over the concrete domain defined as follows:

$$\begin{aligned} &\mathcal{F}[\![.]\!] : \Phi \to (D^\flat \to D^\flat) \\ &\mathcal{F}[\![\varphi]\!]A \triangleq A \cap [\![\varphi]\!]^\flat \end{aligned}$$

We can construct $\mathcal{F}[\![.]\!]$ inductively. First, we define the semantics of terms $\mathcal{T}[\![.]\!] : \mathit{Term} \to (\mathit{Asn} \to \mathbb{U})$ inductively:

$$\begin{aligned} \mathcal{T}[\![x]\!]\rho &= \rho(x) \\ \mathcal{T}[\![f(t_1, \ldots, t_n)]\!]\rho &= [\![f]\!]_F(\mathcal{T}[\![t_1]\!]\rho, \ldots, \mathcal{T}[\![t_n]\!]\rho) \end{aligned}$$

And then the semantics of formulas:

$$\begin{aligned} \mathcal{F}[\![\mathit{true}]\!]A &= A \\ \mathcal{F}[\![\mathit{false}]\!]A &= \{\} \\ \mathcal{F}[\![p(t_1, \ldots, t_n)]\!]A &= \{\rho \in A \mid (\mathcal{T}[\![t_1]\!]\rho, \ldots, \mathcal{T}[\![t_n]\!]\rho) \in [\![p]\!]_P\} \\ \mathcal{F}[\![\neg\varphi]\!]A &= A \setminus \mathcal{F}[\![\varphi]\!]\mathit{Asn} \\ \mathcal{F}[\![\varphi_1 \wedge \varphi_2]\!]A &= \mathcal{F}[\![\varphi_1]\!]A \cap \mathcal{F}[\![\varphi_2]\!]A \\ \mathcal{F}[\![\varphi_1 \vee \varphi_2]\!]A &= \mathcal{F}[\![\varphi_1]\!]A \cup \mathcal{F}[\![\varphi_2]\!]A \end{aligned}$$

Solutions of a FOL Formula

The solutions of φ are given by the greatest fixed point $\mathit{gfp}^{\subseteq}\ \mathcal{F}[\![\varphi]\!]$.

Lemma

$\mathit{gfp}^{\subseteq}\ \mathcal{F}[\![\varphi]\!] = [\![\varphi]\!]^{\flat}$

Similarly to abstract interpretation, we will look for an abstraction to compute more efficiently the set of solutions.

Abstract Propagation

Abstract Domain

Definition

An abstract domain is a lattice $\langle A^\sharp, \sqsubseteq, \sqcup, \sqcap, \bot, \top, \mathcal{F}^\sharp[\![.]\!]\rangle$ such that:

- Every element of $A^\sharp$ is representable in a machine.
- The operations on $A^\sharp$ are efficiently computable.
- $\mathcal{F}^\sharp[\![.]\!]$ is order-preserving.

The concrete and abstract semantics are connected by a Galois connection:

$$\langle \mathcal{P}(X \to \mathbb{U}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A^\sharp, \sqsubseteq\rangle$$

Cartesian Abstraction

As a first approximation of the concrete domain, we take the Cartesian abstraction $X \to \mathcal{P}(\mathbb{U})$ which considers the values of each variable independently.

$$\langle \mathcal{P}(X \to \mathbb{U}), \subseteq \rangle \xleftrightarrow[\alpha_\times]{\gamma_\times} \langle X \to \mathcal{P}(\mathbb{U}), \dot{\subseteq} \rangle$$

$$\alpha_\times(P) \triangleq x \in X \mapsto \{\rho(x) \mid \rho \in P\}$$

$$\gamma_\times(\overline{P}) \triangleq \{\rho \in X \to \mathbb{U} \mid \forall x \in X, \rho(x) \in \overline{P}(x)\}$$

where $\dot{\subseteq}$ is the pointwise set inclusion.

Cartesian Abstraction

We can define the abstract semantics of FOL over $X \to \mathcal{P}(\mathbb{U})$ as follows:

$$\begin{aligned}
&\mathcal{F}^\sharp_\times [\![p(t_1, \ldots, t_n)]\!] \overline{P} \triangleq \\
&\qquad x \in X \mapsto \{ v \in \overline{P}(x) \mid \exists v_1 \in \mathcal{F}^\sharp_\times [\![t_1]\!] \overline{P}[x \mapsto \{v\}], \ldots, v_n \in \mathcal{F}^\sharp_\times [\![t_n]\!] \overline{P}[x \mapsto \{v\}], \\
&\qquad\quad (v_1, \ldots, v_n) \in [\![p]\!]_P \} \\
&\mathcal{F}^\sharp_\times [\![\varphi_1 \wedge \varphi_2]\!] \overline{P} \triangleq \mathcal{F}^\sharp_\times [\![\varphi_1]\!] \overline{P} \cap^\times \mathcal{F}^\sharp_\times [\![\varphi_2]\!] \overline{P} \\
&\mathcal{F}^\sharp_\times [\![\varphi_1 \vee \varphi_2]\!] \overline{P} \triangleq \mathcal{F}^\sharp_\times [\![\varphi_1]\!] \overline{P} \cup^\times \mathcal{F}^\sharp_\times [\![\varphi_2]\!] \overline{P}
\end{aligned}$$

Soundness of $\mathcal{F}^{\sharp}_{\times}[\![.]\!]$

Soundness for gfp

Let $\alpha \circ f \circ \gamma \,\dot{\sqsubseteq}\, \overline{f}$. Then $\mathbf{gfp}^{\leq} f \;\leq\; \gamma(\mathbf{gfp}^{\sqsubseteq} \overline{f})$.

Theorem

The semantics $\mathcal{F}^{\sharp}_{\times}[\![\varphi]\!]$ is sound:

$$\alpha_{\times} \circ \mathcal{F}[\![\varphi]\!] \circ \gamma_{\times} \;\dot{\sqsubseteq}\; \mathcal{F}^{\sharp}_{\times}[\![\varphi]\!]$$

Proof.

By induction over the formula (case of $\wedge$):

$$\begin{array}{rl} & (\alpha_{\times} \circ \mathcal{F}[\![\varphi_1 \wedge \varphi_2]\!] \circ \gamma_{\times})\overline{P} \\ = & \alpha_{\times}(\mathcal{F}[\![\varphi_1]\!]\gamma_{\times}(\overline{P}) \cap \mathcal{F}[\![\varphi_2]\!]\gamma_{\times}(\overline{P})) \\ = & \alpha_{\times}(\mathcal{F}[\![\varphi_1]\!]\gamma_{\times}(\overline{P})) \sqcap \alpha_{\times}(\mathcal{F}[\![\varphi_2]\!]\gamma_{\times}(\overline{P})) \\ \dot{\sqsubseteq} & \mathcal{F}^{\sharp}_{\times}[\![\varphi_1]\!]\overline{P} \sqcap \mathcal{F}^{\sharp}_{\times}[\![\varphi_2]\!]\overline{P} \\ = & \mathcal{F}^{\sharp}_{\times}[\![\varphi_1 \wedge \varphi_2]\!]\overline{P} \end{array}$$

□

Interval Abstract Domain

The abstract domain of interval is
$\mathcal{I}^\sharp \triangleq \langle X \to \mathcal{I}, \dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}, x \in X \mapsto \bot, x \in X \mapsto [-\infty, \infty], \mathbf{C}^\sharp_I[\![.]\!]\rangle$ where
$\dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}$ are pointwise interval operations.

We have the Galois connection:

$$\langle X \to \mathcal{P}(\mathbb{U}), \dot{\subseteq}\rangle \xleftarrow[\overline{\alpha}]{\overline{\gamma}} \langle X \to \mathcal{I}, \dot{\sqsubseteq}\rangle$$
$$\overline{\alpha}(S) \triangleq x \in X \mapsto [min\ S(x), max\ S(x)]$$
$$\overline{\gamma}(R) \triangleq x \in X \mapsto \{c \in \mathbb{U} \mid \lfloor R(x) \rfloor \leq c \leq \lceil R(x) \rceil\}$$

Propagators

In the previous session, we defined:

$$\begin{aligned} \mathbf{C}_I^\sharp[\![x \le y]\!]\sigma \triangleq \\ & \sigma[x \mapsto \sigma(x) \sqcap [-\infty, \lceil \sigma(y) \rceil]] \\ \dot{\sqcap}\ & \sigma[y \mapsto \sigma(y) \sqcap [\lfloor \sigma(x) \rfloor, \infty]] \end{aligned}$$

$\mathbf{C}_I^\sharp[\![x \le y]\!]$ corresponds to the definition of *propagators* in constraint programming.

Propagators

In the previous session, we defined:

$$\begin{aligned}\mathbf{C}_I^\sharp[\![x \leq y]\!]\sigma \;\triangleq\; & \\ & \sigma[x \mapsto \sigma(x) \sqcap [-\infty, \lceil \sigma(y) \rceil]] \\ \dot{\sqcap}\; & \sigma[y \mapsto \sigma(y) \sqcap [\lfloor \sigma(x) \rfloor, \infty]]\end{aligned}$$

$\mathbf{C}_I^\sharp[\![x \leq y]\!]$ corresponds to the definition of *propagators* in constraint programming.

Given a conjunction of constraints such as $x \leq y \wedge y \neq z \wedge z = x/y$, we can compute an overapproximation of the solutions set by:

$$propagate(\rho) \triangleq \mathbf{gfp}_\rho^{\sqsubseteq}\, (\mathbf{C}_I^\sharp[\![x \leq y]\!] \circ \mathbf{C}_I^\sharp[\![y \neq z]\!] \circ \mathbf{C}_I^\sharp[\![z = x/y]\!])$$

By theorems of abstract interpretation, it is a sound solving procedure: it does not discard solutions from the problem.

Abstract Constraint Search

Traditional Constraint Solving

A classic solver in constraint programming:

$\texttt{solve}(\langle X, D, C\rangle)$
$\langle X, D', C\rangle \leftarrow \texttt{propagate}(\langle X, D, C\rangle)$
if D' is an assignment **then**
    return $\{D'\}$
else if D' has an empty domain **then**
    return $\{\}$
else
    $\langle D_1, \ldots, D_n\rangle \leftarrow \texttt{branch}(D')$
    return $\bigcup_{i=0}^{n} \texttt{solve}(\langle X, D_i, C\rangle)$
end if

Abstract Constraint Solving

A solver by abstract interpretation, with $A^\sharp$ an abstract domain:

$\texttt{solve}[\![\varphi]\!](a \in A^\sharp)$
$a \leftarrow \texttt{propagate}[\![\varphi]\!](\texttt{a})$
if $\texttt{split(a)} = \{\texttt{a}\}$ **then**
    return $\{a\}$
else if $\texttt{split(a)} = \{\}$ **then**
    return $\{\}$
else
    $\langle a_1, \ldots, a_n \rangle \leftarrow \texttt{split(a)}$
    return $\bigcup_{i=0}^{n} \texttt{solve}[\![\varphi]\!](a_i)$
end if

- **Conservative extension:** Traditional CP is based on a Cartesian abstraction such as the interval abstract domain.
- **Many abstract domains:** Octagon, Polyhedron, **products**, ...
- An additional abstract function *split* : $A^\sharp \to \mathcal{P}(A^\sharp)$

Hoare and Smyth Lattices

Powerset is not enough...

Let $\langle L, \leq \rangle$ be a lattice.
The powerset completion is $\langle \mathcal{P}(L), \subseteq \rangle$ but..

- Two distinct elements a and b, such that $a \leq_L b$, are not ordered in $\mathcal{P}(L)$ since $\{a\} \not\subseteq \{b\}$.
- We have redundant elements, e.g., if $a \leq_L b$, then the set $\{a, b\}$ contains the redundant element b.

This stems from the fact that the powerset completion views its elements as atomic, and that its ordering is defined regardless of the structure of L.

Down-set and Up-set

A traditional way of dealing with this issue is to take the down-set or up-set completion of the base lattice.

Definition (Down-set and up-set)

Let P be a poset, and $S \subseteq P$. The down-set $\downarrow S$ and up-set $\uparrow S$ are defined by:

$$\downarrow S = \{y \in P \mid \exists x \in S, y \leq x\} \qquad \uparrow S = \{y \in P \mid \exists x \in S, y \geq x\}$$

Let $a \in P$, then we write $\downarrow a$ for $\downarrow\{a\}$ and $\uparrow a$ for $\uparrow\{a\}$. The set of all down-sets of P is denoted $\mathcal{D}(P)$, and the set of all up-sets is denoted $\mathcal{U}(P)$.

Theorem

$\langle \mathcal{D}(P), \subseteq, \cup, \cap, \{\}, P \rangle$ and $\langle \mathcal{U}(P), \supseteq, \cap, \cup, P, \{\} \rangle$ are complete lattices.

But it does not solve the redundancy issue.

Antichains

To overcome this drawback, we consider the antichains of a lattice L.

Definition (Antichain, minimal and maximal elements)

Let $\langle L, \leq \rangle$ be a lattice. An antichain is a set $S \subseteq L$ such that for all pairs of elements $a, b \in S$, we have $a \leq b \Leftrightarrow a = b$. Given a set $Q \subseteq L$, the set of its minimal and maximal elements are defined as follows:

$$Min\ Q = \{x \in Q \mid \forall y \in Q,\ \neg(x >_L y)\}$$
$$Max\ Q = \{x \in Q \mid \forall y \in Q,\ \neg(x <_L y)\}$$

By definition, *Min Q* and *Max Q* are antichains.

Example

Consider the set of sets $S = \{\{0,1\},\{1,2\},\{0\},\{1\}\} \subset \mathcal{P}(\mathbb{Z})$ such that each element in S is ordered by subset inclusion. Then we have $Min\ S = \{\{0\},\{1\}\}$ and $Max\ S = \{\{0,1\},\{1,2\}\}$.

Hoare lattice

We equip the set of antichains of a lattice with two orderings called the *Hoare* and *Smyth* orderings [Plo76; Smy78].

Definition (Hoare construction)

Let $\langle L, \leq \rangle$ be a lattice. Then the Hoare construction $\langle L^H, \leq, \sqcup, \sqcap, \bot, \top \rangle$ is defined as follows:

- $L^H = \{S \in \mathcal{P}_f(L) \mid S \text{ is an antichain in } L\}$,
- $X \leq Y \triangleq \forall y \in Y,\ \exists x \in X,\ x \leq_L y$,
- $X \sqcup Y \triangleq Min\ \{x \sqcup_L y \mid x \in X \wedge y \in Y\}$,
- $X \sqcap Y \triangleq Min\ (X \cup Y)$,
- $\bot \triangleq \{\bot_L\}$ and $\top \triangleq \{\}$.

Smyth lattice

Definition (Smyth construction)

Let $\langle L, \leq \rangle$ be a lattice. Then the Smyth construction $\langle L^S, \leq, \sqcup, \sqcap, \bot, \top \rangle$ is defined as follows:

- $L^S = \{S \in \mathcal{P}_f(L) \mid S \text{ is an antichain in } L\}$,
- $X \leq Y \triangleq \forall x \in X,\ \exists y \in Y,\ x \leq_L y$,
- $X \sqcup Y \triangleq Max\ (X \cup Y)$,
- $X \sqcap Y \triangleq Max\ \{x \sqcap_L y \mid x \in X \wedge y \in Y\}$,
- $\bot \triangleq \{\}$ and $\top \triangleq \{\top_L\}$.

Theorem

Let L be a lattice, then $\langle L^H, \leq \rangle$ and $\langle L^S, \leq \rangle$ are lattices.

Intuitions

Let $\{a, b\}$ be an antichain in the base lattice L.

- For both orderings: $\{a, b\} \leq \{a, c\}$ if $b \leq_L c$.
- For Smyth, an antichain $\{a, b\}$ can be extended with any new element $d \in L$ that is not comparable to a or b, thus obtaining the new antichain $\{a, b, d\}$.
- For Hoare, we can forget about some uninteresting elements—for example inconsistent states—and thus we have $\{a, b\} \leq_H \{a\}$.

Abstract Search

Queuing Strategy

Let A be an abstract domain.

A^H is the structure of the search tree.

Definition (Queuing strategy)

A queuing strategy is a pair of functions $(push, pop)$ defined as follows:

$$push : A^H \times A^H \to A^H$$
$$push(Q, B) \triangleq Q \sqcap_H B$$

$$pop : A^H \to A^H \times A$$
$$pop(Q) \triangleq \begin{cases} (Q, a) & \text{iff } \exists a \in Q,\ |split(a)| > 1 \\ (Q, \bot_A) & \text{otherwise} \end{cases}$$

Abstract Solving

Let $\langle A, \sqsubseteq \rangle$ be an abstract domain with a function $split : A \to A^H$.
The fixpoint form of the constraint solving algorithm is:

$$\begin{aligned} & solve : \Phi \to (A^H \to A^H) \\ & solve[\![\varphi]\!] \triangleq push \circ (id \times (split \circ propagate[\![\varphi]\!])) \circ pop \end{aligned}$$

Theorem

Let $\langle A, \sqsubseteq \rangle$ be an abstract domain with concretization γ_A. Let φ be a formula. If $\gamma_A(a) = \gamma(split(a))$, then

$$\mathbf{gfp}^{\subseteq} \mathcal{F}[\![\varphi]\!] \subseteq \gamma(\mathbf{gfp}^{\sqsubseteq} solve[\![\varphi]\!])$$

with $\gamma(H) \triangleq \bigcup_{a \in H} \gamma_A(a)$.

Conclusion

In-Progress: Abstract Constraint Programming

Collaboration with Bruno Teheux

- Formal theory of propagators using *calculational design*.
- Solvers are fixpoint functions over abstract domains.
- Proofs in Lean/Coq?

- **Context:** In constraint programming, *global constraints* are propagators with dedicated inference algorithms for subproblems, e.g., `alldifferent([`x_1`,...,`x_n`])`.
- Research question: Which global constraints can be generalized into abstract domains?

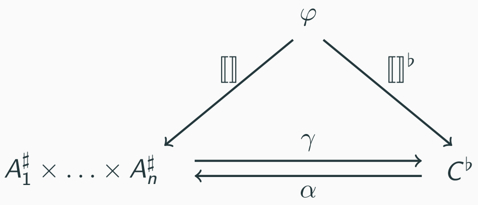
Collaboration with Éric Monfroy

We are working on the *Table abstract domain* generalizing the well-known `table` constraint:

$$\begin{aligned} &(x \geq 4 \land y > 1 \land z < 3) \\ \lor &(x = 1 \land y = 2 \land z = 3) \\ \lor &(x > 1 \land y > 1 \land z > 3) \end{aligned}$$

Perspective: Towards automatic creation of the abstract domain

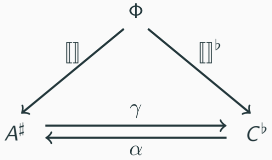
Research question: Given a set of abstract domains and reduced products, how to build the most efficient one to solve a given formula?

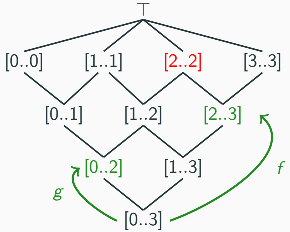


- How to create an appropriate combination of abstract domains for a particular formula?
- "Type inference": In which abstract domain goes each subformula $\varphi_i \in \varphi$?

Conclusion

- Abstract interpretation a "*grand unification theory*" among the fields of constraint reasoning?
- Not there yet, but interesting theory and promising results!





References

[CH78] Patrick Cousot and Nicolas Halbwachs. **"Automatic discovery of linear restraints among variables of a program".** In: *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1978, pp. 84–96.

[Cou20] Patrick Cousot. **"The Symbolic Term Abstract Domain".** In: *TASE* (Dec. 2020). URL: `https://sei.ecnu.edu.cn/tase2020/file/video-slides-PCousot-TASE-2020.pdf`.

[DHK13] Vijay D'Silva, Leopold Haller, and Daniel Kroening. **"Abstract Conflict Driven Learning".** In: *POPL '13*. ACM, 2013, pp. 143–154. DOI: `10.1145/2429069.2429087`.

[DHK14] Vijay D'Silva, Leopold Haller, and Daniel Kroening. **"Abstract satisfaction".** en. In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL '14*. San Diego, California, USA: ACM Press, 2014, pp. 139–150. ISBN: 978-1-4503-2544-8. DOI: 10.1145/2535838.2535868. URL: http://dl.acm.org/citation.cfm?doid=2535838.2535868 (visited on 09/17/2019).

[Pel+13] Marie Pelleau et al. **"A constraint solver based on abstract domains".** In: *VMCAI 13'*. Springer, 2013, pp. 434–454. DOI: 10.1007/978-3-642-35873-9_26.

[Plo76] G. Plotkin. **"A Powerdomain Construction".** In: *SIAM Journal on Computing* 5.3 (1976), pp. 452–487. DOI: 10.1137/0205035.

[Smy78] M. B. Smyth. **"Power domains".** In: *Journal of Computer and System Sciences* 16.1 (1978), pp. 23 –36. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(78)90048-X. URL: http://www.sciencedirect.com/science/article/pii/002200007890048X.