Symbolic Artificial Intelligence (COMP3008)
Lecture 4: SAT Solvers and CSP
20 October 2025

Daniel Karapetyan
daniel.karapetyan@nottingham.ac.uk

Propositional Logic and SAT

Propositional Logic

- Propositional logic is a subset of FOL
- It is 'FOL without the domain of discourse'
- It includes zero-arity predicates, operators $\wedge$, $\vee$ and $\neg$ and brackets
- Note that $\rightarrow$ and $\leftrightarrow$ are also included as they can be expressed via $\wedge$, $\vee$ and $\neg$
- Example of a propositional logic formula: $(P \vee Q) \wedge \neg R$

Propositional logic vs FOL

If the domain of discourse is finite, propositional logic is as expressive as FOL

- You have seen in Worksheet 3 how we can expand quantifiers and replace a function with multiple constants
- We can also replace a constant with multiple zero-arity predicates

  For example, an integer $1 \leq i \leq 3$ can be replaced with three predicates: i_1, i_2 and i_3, with a requirement that exactly one of is TRUE

Let us focus on propositional logic

Satisfiability vs. entailment

Recall that a knowledge-based system may either

- Seek proofs of entailment (i.e., that a given hypothesis is a logical consequence of the knowledge base)
- Seek an interpretation (model) that satisfies the knowledge base

An algorithm for establishing entailment can be used to establish satisfiability and vice versa:

$$\text{KB} \models \alpha \qquad \text{if and only if} \qquad (\text{KB} \wedge \neg\alpha) \text{ is unsatisfiable}$$

SAT Solvers

Nevertheless, it makes sense to have algorithms designed specifically for searching for interpretations

A standard form for such a problem is called SAT:

Given a propositional formula in Conjunctive Normal Form (CNF), find an interpretation that satisfies it or prove that it is unsatisfiable

- The SAT problem is NP-complete
- In practice, SAT solvers are extremely efficient
- We will study the main SAT algorithm (DPLL) in another lecture
- Some general-purpose solvers actually convert problems into SAT to exploit this efficiency

Constraint Satisfaction

Constraint Satisfaction Problem

One example of the problem that is often *reduced* to SAT is the *Constraint Satisfaction Problem* (CSP)

CSP is a powerful extension of SAT for modelling mathematical problems

CSP is effectively another language to describe knowledge:

- For finite domains, equivalent to SAT in expressive power
- Rich: complex knowledge can be formulated compactly
- Benefits from effective solvers

CSP definition

In CSP, you are given

- A set of variables $x_1, x_2, \ldots, x_n$
- The domains $D_1, D_2, \ldots, D_n$
  - The domains can be finite or infinite
- A set of constraints C

The problem is to find the values of $x_i \in D_i$ for each i such that all the constraints C are satisfied, or prove that no such values exist

CSP constraints

A CSP constraint is a relation between one or several variables

The set of variables used in the relation is called *scope*

Informally speaking, a constraint tells for each specific combination of the scope variables if this combination satisfies it

A constraint can be defined compactly (e.g., using arithmetic) or in a tabular form

CSP constraints – ‘not-equals’

‘Not-equals’ constraint:

- Scope: $\{x_1, x_2\}$
- Compact definition: satisfied if x_1 is not equal to x_2
- Tabular definition assuming $D_1 = D_2 = \{1, 2, 3\}$:

$x_1 \setminus x_2$	1	2	3
1	−	+	+
2	+	−	+
3	+	+	−

CSP constraints – Alldiff

Alldiff is a generalisation of ‘not-equals’:

- Scope: $\{x_1, x_2, \ldots, x_k\}$
- Compact definition: satisfied if $x_i \neq x_j$ for every $i \neq j \in \{1, 2, \ldots, k\}$
- You can see an example of a tabular definition for $k = 2$ in slide 11

CSP constraints – arithmetic

A constraint can be defined using arithmetic, e.g.:

- Variables x_1 and x_2 with domains $D_1 = D_2 = \{1, 2, 3\}$
- Compact definition: satisfied if $2x_1 + x_2 \leq 5$
- Tabular definition:

$x_1 \setminus x_2$	1	2	3
1	+	+	+
2	+	−	−
3	−	−	−

Ways to define constraints

- Tabular definitions only work for finite domains
- Tabular definitions are usually impractical
- We tend to use compact descriptions
  - Have to have a deterministic algorithm to test if a specific assignment of values to the scope variables satisfies the constraint
  - This algorithm is expected to be quick

Constraint Satisfaction Solvers

CSP vs SAT

- CSP with finite domains has the same expressive power as SAT
  - Translation in either direction is possible
- CSP with infinite domains cannot be translated into SAT
- CSP formulations are typically much more compact and intuitive

Analogy between CSP and FOL (example 1)

What's the CSP equivalent of the following FOL sentence:

$$A \wedge B \tag{1}$$

Analogy between CSP and FOL (example 1)

What's the CSP equivalent of the following FOL sentence:

$$A \wedge B \tag{1}$$

CSP:

$$A \wedge B = \text{True} \tag{2}$$

$$A, B \in \{0, 1\} \tag{3}$$

Analogy between CSP and FOL (example 2)

What's the CSP equivalent of the following FOL sentence:

$$\exists x.A(x) \tag{4}$$

for $\mathcal{D} = \{1, 2, \ldots, n\}$

Analogy between CSP and FOL (example 2)

What's the CSP equivalent of the following FOL sentence:

$$\exists x.A(x) \tag{4}$$

for $\mathcal{D} = \{1, 2, \ldots, n\}$

CSP:

$$A_1 \vee A_2 \vee \cdots \vee A_n = \text{True} \tag{5}$$

$$A_i \in \{0, 1\} \qquad \forall i \in 1, 2, \ldots, n \tag{6}$$

Analogy between CSP and FOL (example 3)

What's the CSP equivalent of the following FOL sentence:

$$\forall x.A(x) \tag{7}$$

for $\mathcal{D} = \{1, 2, \ldots, n\}$

Analogy between CSP and FOL (example 3)

What's the CSP equivalent of the following FOL sentence:

$$\forall x.A(x) \tag{7}$$

for $\mathcal{D} = \{1, 2, \ldots, n\}$

CSP:

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n = \text{True} \tag{8}$$

$$A_i \in \{0, 1\} \qquad \forall i \in 1, 2, \ldots, n \tag{9}$$

Summary

- CSP is designed specifically for problems where we need to find an interpretation (model)
- CSP is not as expressive as FOL; it is unlikely to be a good choice, e.g., for theorem proving
- CSP formulations include more information about the problem structure than SAT formulations
  - This enables more efficient reasoning
- Some solvers simply translate CSP into SAT and use a SAT solver, but usually will supply the SAT solver with hints about the problem structure