Theory of Answer Set Programming

Definite/Positive Programs in the Language of Clingo

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Represent definite and positive programs in the language of clingo and using clingo to find the stable models

Objective

Represent definitions in the language of clingo

Positive Programs in the Language of clingo

Terms in clingo Language

- **Integers, symbolic constants, and variables**
  - 1, 2, 3, a, b, X
- **$t_1 \circ t_2$ where $\circ$ is an arithmetic operation, and t_1, t_2 are terms**
  - 3*4, 3+4, 3**X+2*3
- **$|t|$ where t is a term**
  - |-3|

Terms that do not contain variable are called ground.

NOTE

Variables are capitalized; symbolic constants are not.

Value of Ground Term

The value of a ground term is defined recursively.

- If t is an integer or a symbolic constant then the only value of t is t itself.
- If t is $t_1 \circ t_2$, where $\circ$ is an arithmetic operation, then the value of t is the integer $n_1 \circ n_2$, where the integer n_1 is a value of t_1, and the integer n_2 is a value of t_2.
- If t is $|t_1|$ then the value of t is an integer $|n_1|$, where the integer n_1 is a value of t_1.

CLINGO terms	Algebraic notation
$m*n$	$m \bullet n$
m/n	$\lfloor m/n \rfloor$
$m\backslash n$	$m - n \bullet \lfloor m/n \rfloor$
$m**n$	$\lfloor m^n \rfloor$

Positive clingo Program

An atom in a clingo program is a symbolic constant followed by a list of terms

A comparison is a pair of terms separated by comparison operators =, !=, <, >, <=, >=

- True: $10>7$, $a > 7$, $c > a$

Positive clingo rules are either

- $H_1, \ldots, H_m \quad (m \geq 1)$ or
- $H_1, \ldots, H_m :- B_1, \ldots, B_n \quad (m, n \geq 0)$

where $H_1, \ldots, H_m, B_1, \ldots, B_n$ are atoms and comparisons

Simple Example

Data

Country	France	Germany	Italy	United Kingdom
Population (mln)	65	83	61	64

Rule:

- A country C is large if the population size of C is S_1 , the population size of the UK is S_2 , and $S_1 > S_2$.

This is not a command; it is a declarative sentence explaining how we understand “large country.”

Simple Example, cont'd

Data

Country	France	Germany	Italy	United Kingdom
Population (mln)	65	83	61	64

size(france,65). size(germany,83). size(italy,61). size(uk,64).

Rule:

A country C is large if the population size of C is S_1, the population size of the UK is S_2 , and $S_1 > S_2$.

large(C) :- size(C,S1), size(uk ,S2), S1 > S2.

Stable model (a.k.a. answer set):

size(france,65) size(germany,83) size(italy,61) size(uk,64) large(france) large(germany)

Rule Instances

An instance of a rule is obtained from the rule by substituting specific values for all its variables

- The values come from the set S of symbolic constants and the set Z of integers

Example:

Rule: size(france,65). size(germany,83). size(italy,61). size(uk,64).

Instances: size(france,65). size(germany,83). size(italy,61). size(uk,64).

Rule: `large(C) :- size(C,S1), size(uk,S2), S1 > S2.`

Instances: `large(`v_0`) :- size(`v_0`, `v_1`), size(uk, `v_2`), `v_1` > `v_2`.`

for all $v_0, v_1, v_2 \in \boldsymbol{S} \cup \boldsymbol{Z}$, where

$\boldsymbol{S}$ **= {france, germany, italy, uk, large, size}**

Propositional Image of clingo Programs: Definition

The propositional image of a clingo program consists of the instances of its rules rewritten as propositional formulas.

To rewrite a ground rule as a formula,

- **replace the symbol :- and all commas in the head and the body by propositional connectives as in the table shown;**
- **replace each comparison $t_1 \prec t_2$ in the head and in the body by T if it is true, and by $\bot$ if it is false; and**
- **replace the head of the rule by $\bot$ if it is empty, and replace the body by T if it is empty.**

CLINGO rules	Propositional rules
:-	$\leftarrow$
comma in the body	$\wedge$
comma in the head	$\vee$
not	$\neg$
#false	$\bot$
#true	$\top$

Example

Propositional image of

```
size(france,65). size(germany,83). size(italy,61). size(uk,64).
large(C) :- size(C,S1), size(uk,S2), S1 > S2.
```

is

$$size(france, 65),\quad size(germany, 83),\quad size(italy, 61),\quad size(uk, 64).$$

$$large(v_0) \leftarrow size(v_0, v_1) \wedge size(uk, v_2) \wedge \top \qquad \text{if } v_1 > v_2$$
$$large(v_0) \leftarrow size(v_0, v_1) \wedge size(uk, v_2) \wedge \bot \qquad \text{otherwise}$$

for all $v_0,\ v_1, v_2$ `in` $\boldsymbol{S}$ `U` $\boldsymbol{Z}$`.`

Example, cont'd

$size(france, 65),\quad size(germany, 83),\quad size(italy, 61),\quad size(uk, 64).$

$large(v_0) \leftarrow size(v_0, v_1) \wedge size(uk, v_2) \wedge \top$ **if** $v_1 > v_2$

~~$large(v_0) \leftarrow size(v_0, v_1) \wedge size(uk, v_2) \wedge \bot$~~ **otherwise**

is simplified as

$size(france, 65),\ size(germany, 83),\ size(italy, 61),\ size(uk, 64).$

$large(v_0) \leftarrow size(v_0, v_1) \wedge size(uk, v_2) \wedge \top$ **if** $v_1 > v_2$

64

65

83

and further as

$size(france, 65),\ size(germany, 83),\ size(italy, 61),\ size(uk, 64),$

$large(france) \leftarrow size(france, 65) \wedge size(uk, 64)$

$large(germany) \leftarrow size(germany, 83) \wedge size(uk, 64)$

EXERCISE

Running in clingo

```
% clingo myfile.lp
```

instructs Clingo to find a stable model of the program myfile.lp.

(a) The population of Russia in 2015 was 142 million. Add this fact to your file and check how this affects the output of clingo.

(b) Instead of comparing countries with the United Kingdom, let us define “large” as having more than 500 million inhabitants. Modify your file accordingly, and check how this change affects the output of clingo.

Expressing Definitions in Positive Programs

EXERCISE

Consider the rule

```
child (X,Y)   :- parent (Y,X).
```

(a) How would you translate this rule into English?

(b) If we run CLINGO on the program consisting of the rule above and the rules

```
parent (ann,bob).  parent (bob,carol).  parent (bob,dan).
```

then what would be the stable model?

EXERCISE

```
parent(ann,bob).   parent(bob,carol).   parent(bob,dan)
```

How would you define the predicate grandparent/2 **in terms of** parent/2**?**

```
parent(ann,bob).   parent(bob,carol).   parent(bob,dan)
```

How would you define the predicate sibling/2 **in terms of** parent/2**?**

EXERCISE

Definition using Multiple Rules

- **Sometimes the definition of a predicate consists of several rules.**
  - **For example, the pair of rules**

```
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

defines `parent/2` **in terms of** `father/2` **and** `mother/2`.

EXERCISE

A predicate can be defined recursively. In a recursive definition, the defined predicate occurs not only in the heads of the rules but also in some of the bodies.

- The definition of `ancestor/2` in terms of `parent/2 is` an example:

```
ancestor (X,Y) :- parent(X,Y).
ancestor (X,Z) :- ancestor(X,Y) ancestor (Y,Z).
```

```
parent (ann,bob).  parent (bob,carol).  parent(bob, dan).
```

Lecture Wrap-Up