1 Definitions

Definition 1 (Formula).

– $\emptyset$ is a *formula*
– $T \in D$, where T is a variable, a ground term or an arithmetic term, and D is a set of ground terms, is a *formula*,
– $t_1 \diamond t_2$, where t_1 and t_2 are terms and $\diamond \in \{=, \neq, \prec, \preceq\}$, is a *formula*, and
– if A and B are formulas then $(A \wedge B)$, $(A \vee B)$, and $\neg A$ are *formulas*.

Definition 2 (Empty Formula).

– $\emptyset$ *is an* empty formula
– *if A and B are empty formulas then $(A \wedge B)$, $(A \vee B)$, and $\neg A$ are* empty formulas.

In what follows any empty formula is interpreted as **false**.

Definition 3 (Primitive Formula).

– $T \in D$, where T is a variable, a ground term or an arithmetic term, and D is a set of ground terms, is a *primitive formula*,
– $t_1 \diamond t_2$, where t_1 and t_2 are terms and $\diamond \in \{=, \neq, \prec, \preceq\}$, is a *primitive formula*, and
– if A and B are formulas then $\neg A$ is a *primitive formula*.

Definition 4 (Primitive Conjunction).
A formula $\mathcal{F}$ is a *primitive conjunction* if it is of the form $G_1 \wedge \cdots \wedge G_n$, where $G_1, \ldots, G_n$ are primitive formulas.

Definition 5 (Arithmetic Variable).
A variable X occuring in a primitive conjunction $\mathcal{C} = G_1 \wedge \cdots \wedge G_n$ is called arithmetic with respect to $\mathcal{C}$ if one of the following conditions holds:

– X occurs in an arithmetic term containing at least one arithmetic operation.
– one of G_i is of the form $X \in D$, where D is a range of natural numbers.
– one of G_i is of the form $X \diamond Y$ or $Y \diamond X$, where Y is an arithmetic variable and $\diamond \in \{\prec, <, \preceq, =\}$.

Definition 6 (Arithmetic Term). A term t occuring in a primitive conjunction $\mathcal{C} = G_1 \wedge \cdots \wedge G_n$ is called *arithmetic* with respect to $\mathcal{C}$ if one of the following conditions holds:

1. t is a number
2. t contains an arithmetic operation ('+','-', or '*').
3. t is not a record and all variables in t are arithmetic with respect to $\mathcal{C}$

Definition 7 (Primitive Arithmetic Constraint). A primitive constraint G occurring in a primitive conjunction $\mathcal{C} = G_1 \wedge \cdots \wedge G_n$ is called *arithmetic* with respect to $\mathcal{C}$ if one of the following conditions holds:

– G is of the form $T \in D$, where T is an arithmetic term with respect to $\mathcal{C}$ and D is of the form $n1..n2$.
– G is of the form $t_1 \diamond t_2$, where both t_1 and t_2 are arithmetic terms with respect to $\mathcal{C}$.

2 Algorithms

Algorithm 1: ExpandSolve

Input: Formulas $\mathcal{F}$, $\mathcal{TODO}$, $\mathcal{C}$, such that $\mathcal{F} \wedge \mathcal{TODO} \wedge \mathcal{C}$ is non-empty
Output: **True** if $\mathcal{F} \wedge \mathcal{TODO} \wedge \mathcal{C}$ is satisfiable and **false** otherwise

if $\mathcal{F} = \emptyset$ **then**
  if $\mathcal{TODO} = \emptyset$ **then**
    return Solve(Simplify($\mathcal{C}$),$maxint$)
  else
    Let $\mathcal{TODO}$ be $G_1 \wedge \cdots \wedge G_n$
    return *ExpandSolve(*G_1,$G_2 \wedge \cdots \wedge G_n$,$\mathcal{C}$*)*
else if $\mathcal{F} = \neg(A \wedge B)$ **then**
  ExpandSolve($\neg A \vee \neg B, \mathcal{TODO}, \mathcal{C}$)
else if $\mathcal{F} = \neg(A \vee B)$ **then**
  ExpandSolve($\neg A \wedge \neg B, \mathcal{TODO}, \mathcal{C}$)
else if $\mathcal{F} = A \vee B$ **then**
  if *ExpandSolve(*$A, \mathcal{TODO}, \mathcal{C}$*)=**false*** **then**
    return ExpandSolve($B, \mathcal{TODO}, \mathcal{C}$)
  else
    return true
else if $\mathcal{F} = A \wedge B$ **then**
  return ExpandSolve($A, B \wedge \mathcal{TODO}, \mathcal{C}$)
else
  return ExpandSolve($\emptyset, \mathcal{TODO}, \mathcal{F} \wedge \mathcal{C}$)
return true

Algorithm 2: Simplify

Input: Primitive conjunction $G_1 \wedge \cdots \wedge G_n$, the upper limit for natural numbers
Output: Primitive conjunction $G_1 \wedge \cdots \wedge G_m$ after simplification

$\mathcal{C} := G_1 \wedge \cdots \wedge G_n$
foreach G_i *in* $G_1 \wedge \cdots \wedge G_n$ **do**
  if G_i *is of the form* $T \in D$*, and T is a ground term* **then**
    if *T is in D* **then**
      Remove G_i from $\mathcal{C}$
    else
      return false
  if G_i *is of the form* $X \in D$*, X is an arithmetic variable in* $\mathcal{C}$*, and D does not contain a number* **then**
    return false
  if G_i *is of the form* $T \in D$*, T is an arithmetic term with at least one operation, and D does not contain a number* **then**
    return false
  if G_i *is of the form* $t_1 \diamond t_2$ *and both* t_1 *and* t_2 *are ground terms* **then**
    Let G'_i be obtained from G_i where $<, \leq$ replaced with $\prec, \preceq$ respectively.
    if G'_i *is true* **then**
      Remove G_i from $\mathcal{C}$
    else
      return false
  if G_i *is of the form* $t_1 \diamond t_2$*, where one of* t_1 *and* t_2 *is an arithmetic term with at least one operation, number, arithmetic variable; and another one is a symbolic term (a string constant or a term built from a functional symbol), and diamond is either* $=, <, \leq, \prec,$ *or* $\preceq$ **then**
    return false
  return $\mathcal{C}$

Algorithm 3: Split

Input: A clingcon rule of the form $r(X_1, \ldots X_n) :- BODY$
Output: A collection of clingcon rules R

Let $BODY$ be $A_1, A_2, \ldots A_m$. Let $\mathcal{G}$ be undirected graph with m nodes $N_1, \ldots N_m$, such that there is an edge between N_i and N_j iff atoms A_i and A_j share a common variable. Let $C_1, \ldots, C_k$ be connected components of $\mathcal{G}$.
$R := \emptyset$
foreach *connected component* C_i *in* $C_1, \ldots, C_k$ **do**
    Let C_i consists of nodes $N_{i_1}, \ldots N_{i_t}$. Let $X_1, \ldots, X_p$ be all variables in $A_{i_1}, \ldots A_{i_t}$.
    $R := R \cup r_i :- A_{i_1}, \ldots A_{i_t}$
$R := R \cup r :- r_1, \ldots, r_k$.
$R := R \cup :- not\ r$.

Algorithm 4: Solve

Input: Primitive conjunction $G_1 \wedge \cdots \wedge G_n$, the upper limit for natural numbers $maxint$.

Output: **true** if $G_1 \wedge \cdots \wedge G_n$ is satisfiable and **false** otherwise.

`/* Build rules for arithmetic constraints */`

Π_{prolog} :=: $-use\_module(library(clpfd))$.
$BODY := \mathbf{true}$
foreach *primitive arithmetic* G_i *of the form* $t_1 \diamond t_2$ **do**
  Replace $\prec, =, ! =, <=$ with $\# <, \# =, \# =, \# =<$ respectively
  $BODY := BODY \wedge G_i$
foreach *primitive arithmetic* G_i *of the form* $(t \in D)$*,where* D *is of the form* $[n1..n2]$ **do**
  $BODY := BODY \wedge t \; in \; n_1..n_2$
foreach *primitive arithmetic* G_i *of the form* $\neg(t \in D)$*,where* D *is of the form* $[n1..n2]$*, and* g_i *is an unique label for* G_i **do**
  $\Pi_{prolog} := \Pi_{prolog} \cup g_i(t) : -n\# > n_2. \cup g_i(t) : -n\# < n_2.$
  $BODY := BODY \wedge g_i(t)$
foreach *primitive arithmetic* G_i *of the form* $(t \in D)$ *and* $\neg(t \in D)$ *, where* D *is not of the form* $[n1..n2]$ **do**
  Remove all symbolic terms from D
  Add the following rule to Π_{prolog}: (d is an unique label for D)
  `set_d(X):-member(X,[t1,...tn]).`
foreach *primitive non-arithmetic* G_i *of the form* $\neg(t \in D)$*:* **do**
  $BODY := BODY \wedge not \; set_d(t)$
foreach *primitive non-arithmetic* G_i *of the form* $t \in D$*:* **do**
  $BODY := BODY \wedge set_d(t)$
foreach *primitive non-arithmetic* G_i *of the form* $t_1 \diamond t_2$*:* **do**
  $BODY := BODY \wedge G_i$
foreach *arithmetic variable* X *in* $BODY$ **do**
  $BODY := integer(X) \wedge BODY$
Let $Y_1, \ldots Y_n$ be the set of all variables in $BODY$
$\mathcal{R} := p : -BODY$
$\Pi_{prolog} := \Pi_{prolog} \cup \mathcal{R}$
if Π_{prolog} *outputs 'yes' for query ?-p* **then**
  return true
else
  return false