

RDL

Rewrite and Decision procedure Laboratory

Alessandro Armando

MRG-Lab

DIST, University of Genova



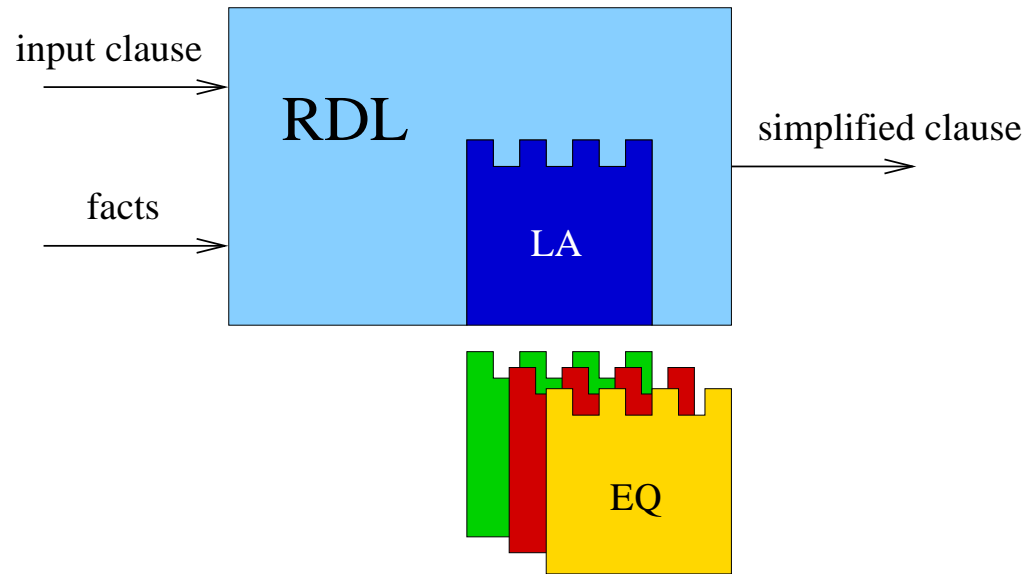
CALCULEMUS Autumn School

Pisa, Sept 30 – Oct 1, 2002

Introduction

- **RDL** simplifies clauses in a quantifier-free first-order logic with equality,
- is based on CCR,
- inherits soundness and termination from CCR,
- allows for the plug&play integration of decision procedures (as long as they comply with some precisely specified requirements), and
- uses *lemma speculation* techniques which extend the scope of the available decision procedures.

RDL: User's View



Example:

Input clause:

$$\{\neg list(l), \neg a \leq \min(l), \neg 0 < k, a < \max(l) + \text{abs}(k)\}$$

Input Facts: $0 < N \rightarrow \text{abs}(N) = N$

$$list(X) \rightarrow \min(X) \leq \max(X)$$

Output: $\{true\}$

Basic Reasoning Specialists

The following basic reasoning specialists are available in the current version of **RDL**:

- **eq**: for the quantifier-free theory of equality (based on Shostak's *congruence closure* algorithm),
- **1a**: for quantifier-free Presburger Arithmetic (based on *Fourier-Motzkin elimination method*), and
- **eq_1a**: for the combination of the previous two theories (obtained by using Nelson & Oppen's combination method).

Lemma Speculation

The following lemma speculation mechanisms are available in the current version of **RDL**:

- **aug**: augmentation, lemmas are instances of available facts.
 - **Pros**: generic (i.e. works for all interpreted symbols)
 - **Cons**: lemmas must be provided by the user
- **aff**: affinization, lemmas are generated on-the-fly
 - **Pros**: fully-automatic
 - **Cons**: generates facts about multiplication over the integers only
- **aug_aff**: combination of the previous two.

Affinization

Problem: Determine the unsatisfiability of

$$\{a < 0, \quad b \geq -a, \quad a * (a + b) \geq 1\}$$

over the integers using a decision procedure for Linear Arithmetic.

Affinization

Problem: Determine the unsatisfiability of

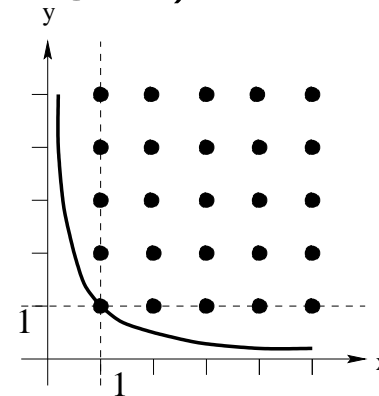
$$\{a < 0, b \geq -a, a * (a + b) \geq 1\}$$

over the integers using a decision procedure for Linear Arithmetic.

Solution:

1. *Affinize hyperbolic inequalities (over the integers):*

$$X * Y \geq 1 \leftrightarrow \left(\begin{array}{c} X \geq 1 \wedge Y \geq 1 \\ \vee \\ X \leq -1 \wedge Y \leq -1 \end{array} \right)$$



2. *Compile into lemmas:*

$$X \geq 1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \geq 1)$$

2. *Compile into lemmas:*

$$X \geq 1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \geq 1)$$

$$X \leq -1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \leq -1)$$

2. *Compile into lemmas:*

$$X \geq 1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \geq 1)$$

$$X \leq -1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \leq -1)$$

⋮

2. *Compile into lemmas:*

$$\begin{aligned} X \geq 1 &\rightarrow (X * Y \geq 1 \leftrightarrow Y \geq 1) \\ X \leq -1 &\rightarrow (X * Y \geq 1 \leftrightarrow Y \leq -1) \\ &\vdots \end{aligned}$$

3. Use lemmas as rewrite rules to “reduce” the non-linearities, e.g.:

$$\{a < 0, \quad b \geq -a, \quad a * (a + b) \geq 1\}$$

2. Compile into lemmas:

$$X \geq 1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \geq 1)$$

$$X \leq -1 \rightarrow (X * Y \geq 1 \leftrightarrow Y \leq -1)$$

⋮

3. Use lemmas as rewrite rules to “reduce” the non-linearities, e.g.:

$$\{a < 0, \quad b \geq -a, \quad a * (a + b) \geq 1\}$$

⇓

if $a \leq -1$

$$\{a < 0, \quad b \geq -a, \quad a + b \leq -1\}$$

Specifying the Reasoning Specialist

The reasoning specialist to apply can be specified by means of the following syntax:

$$RS ::= DP \mid LS(DP)$$
$$DP ::= eq \mid la \mid eq_la$$
$$LS ::= aug \mid aff \mid aug_aff$$

Examples: `la,`
`aug(la),`
`aff(eq_la),`
`aug_aff(eq_la),`
`⋮`

RDL problems description

```
description( Pb,          % Problem name
             Author,     % Problem author
             Note  ).    % Problem description
```

```
input( Pb,          % Problem name
       CL  ).      % Input clause
```

```
expected_output( Pb,          % Problem name
                 RS,          % Reasoning specialist used
                 ORD,         % Ordering used
                 EXP  ).      % Expected clause
```

RDL problems description (cont'd)

```
fact( Pb,           % Problem name
      Lb,           % Lemma name
      Cs,           % Lemma conditions
      Concl ).     % Lemma conclusion

pred_sym( Pb,       % Problem name
          P( _, ..., _ ) ). % Predicate arity
```

RDL interface predicates

To load the file `F` issue

```
load_problems_file(F).
```

RDL interface predicates

To load the file F issue

```
load_problems_file(F).
```

To run **RDL** against problem Pb using the reasoning specialist RS issue:

```
run(RS,Pb).
```

Zhang's example: zhang

Given the lemma

$$A \neq 0 \Rightarrow \text{rem}(A * B, A) = 0$$

then the following formula holds

$$x * y = y * z \wedge \text{rem}(y * z, x) \neq 0 \Rightarrow x = 0$$

RDL proves the validity of this example in 10 msec using [eq.](#)

Zhang's example: zhang_1a

Given the lemma:

$$A > 0 \Rightarrow \text{rem}(A * B, A) = 0$$

then the following formula holds over integers,

$$\text{rem}(x * y, x) \neq 0 \Rightarrow x \leq 0$$

RDL proves the validity of this example in 10 msec using [1a](#).

Boyer and Moore's example: `bm94`

Given the lemma:

$$\mathit{min}(A) \leq \mathit{max}(A)$$

then the following formula holds over integers,

$$l \geq 0 \wedge k > 0 \wedge l \leq \mathit{min}(b) \Rightarrow l < \mathit{max}(b) + k$$

RDL proves the validity of this example in 10 msec using `aug(1a)`.

1 step of augmentation is applied.

Kapur and Nie's example: `kapur_nie_11`

Given the lemmas:

$$\begin{aligned} \max(A, B) = A &\Rightarrow \min(A, B) = B \\ p(C) &\Rightarrow f(C) \leq g(C) \end{aligned}$$

then the following formula holds over integers,

$$\begin{aligned} (p(x) \wedge z \leq f(\max(x, y)) \wedge 0 < \min(x, y) \wedge \\ x \leq \max(x, y) \wedge \max(x, y) \leq x) \\ \Rightarrow z < g(x) + y \end{aligned}$$

RDL proves the validity of this example in 60 msec using `aug(1a)`.
2 steps of augmentation are applied.

Boyer and Moore's example: `bm102`

Given the lemmas:

$$0 < A \Rightarrow B \leq A * B$$

$$0 < ms(C)$$

then the following formula holds over integers,

$$ms(c) + ms(d)^2 + ms(b)^2 < \\ ms(c) + ms(b)^2 + 2 * ms(d)^2 * ms(b) + ms(d)^4$$

RDL proves the validity of this example in 30 msec using `aug(1a)`.

9 steps of augmentation are applied.

Bjorner's non-linear example: `nl-example-bjorner`

The following formula holds over integers,

$$x > 0 \wedge y > 0 \wedge z > 0 \Rightarrow x * y * z + x * y * z * z > 0$$

RDL proves the validity of this problem in 10 msec using `aff(1a)`.

3 steps of affinization are applied.

Boyer and Moore's example: `nl-1-termination`

This example differs from the previous Boyer and Moore's example in that the lemma about multiplication is not provided here to the augmentation process.

We only give the lemma encoding the positivity of ms :

$$0 < ms(A)$$

The formula

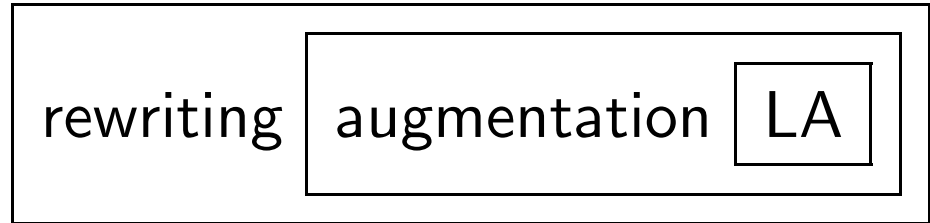
$$ms(c) + ms(d)^2 + ms(b)^2 < \\ ms(c) + ms(b)^2 + 2 * ms(d)^2 * ms(b) + ms(d)^4$$

is proved to be valid in 20 msec using `aug_aff(1a)`.

6 steps of augmentation and affinization are applied.

State of the art

- **AcI2/NQTHM** and **Tecton**:



State of the art

- **AcI2/NQTHM** and **Tecton**:

rewriting

augmentation

LA

- **RDL**:

rewriting

lemma speculation

EQ, LA, EQ+LA, . . .

State of the art

- **AcI2/NQTHM** and **Tecton**:

rewriting

augmentation

LA

- **RDL**:

rewriting

lemma speculation

EQ, LA, EQ+LA, . . .

- **Simplify**:

matching algorithm

EQ+LA

State of the art

- **AcI2/NQTHM** and **Tecton**:

rewriting

augmentation

LA

- **RDL**:

rewriting

lemma speculation

EQ, LA, EQ+LA, . . .

- **Simplify**:

matching algorithm

EQ+LA

- **SVC** and **STeP**:

Propositional/FOL Reasoner

EQ+LA+. . .

State of the art

- **AcI2/NQTHM** and **Tecton**:

rewriting

augmentation

LA

- **RDL**:

rewriting

lemma speculation

EQ, LA, EQ+LA, . . .

- **Simplify**:

matching algorithm

EQ+LA

- **SVC** and **STeP**:

Propositional/FOL Reasoner

EQ+LA+. . .

- **PVS**:

rewriting

EQ+LA+. . .

instantiation of lemmas

Preliminary comparison

System	Problem type			
	eq	la	aug(la)	aug_aff(la)
RDL	100%	100%	100%	100%
<i>Simplify</i>	100%	100%	100%	6%
SVC	100%	100%	0%	11%

where:

- **eq**: ground equality problems (8 pbs)
- **la**: linear arithmetic problem (10 pbs)
- **aug(la)**: LA problem with user supplied facts (8 pbs)
- **aug_aff(la)**: non LA problem with user supplied facts (35 pbs)