Aalto University
School of Science

Improving the Normalization of Weight Rules in Answer Set Programs

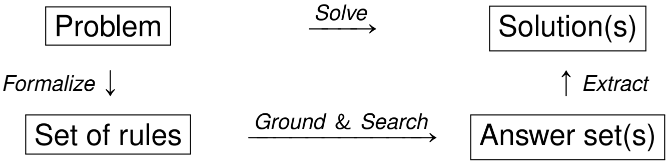
Jori Bomanson, Martin Gebser, and Tomi Janhunen

Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University

JELIA, Madeira, Portugal, September 24, 2014

Background

- Answer set programming (ASP) features a rule-based syntax subject to answer-set semantics.



Different Types of Rules

We consider propositional answer set programs containing:

- Normal rules:

  $a \leftarrow b,\ c,\ \text{not}\, d,\ \text{not}\, e$

- Cardinality rules:

  $a \leftarrow 3 \leqslant \{b,\ c,\ d,\ \text{not}\, e,\ \text{not}\, f\}$

- Weight rules:

  $a \leftarrow 6 \leqslant [b = 2, c = 4, d = 3, e = 3, f = 1, g = 4]$

Objectives:

- Rewrite weight rules using normal rules
- Complement back-ends lacking weight rule support
- Improve efficiency of nogood recording

Example of Normalization

$$a \leftarrow 3 \leqslant \{b,\ c,\ d,\ \mathrm{not}\ e,\ \mathrm{not}\ f\}$$

$$\longmapsto$$

$$a \leftarrow b, c, d.$$
$$a \leftarrow b, c, \mathrm{not}\ e.$$
$$a \leftarrow b, c, \mathrm{not}\ f.$$
$$a \leftarrow b, d, \mathrm{not}\ e.$$
$$a \leftarrow b, d, \mathrm{not}\ f.$$
$$a \leftarrow b, \mathrm{not}\ e, \mathrm{not}\ f.$$
$$a \leftarrow c, d, \mathrm{not}\ e.$$
$$a \leftarrow c, d, \mathrm{not}\ f.$$
$$a \leftarrow c, \mathrm{not}\ e, \mathrm{not}\ f.$$
$$a \leftarrow d, \mathrm{not}\ e, \mathrm{not}\ f.$$

Related Work

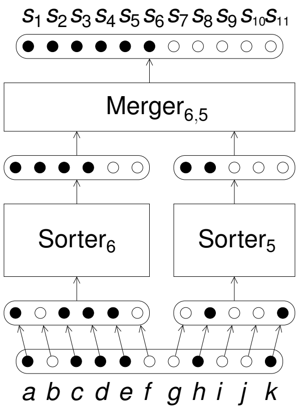
- Eén and Sörensson, JSAT'06
  - Translation of Pseudo-Boolean to sorting networks to SAT
- Bailleux, Boufkhad, and Roussel, SAT'09
  - Polynomial Watchdog translation using tares
- Codish, Fekete, Fuhs, and Schneider-Kamp, TACAS'11
  - Optimal base problem and algorithm(s)
- Bomanson and Janhunen, LPNMR'13
  - Merging and sorting for normalizing cardinality rules

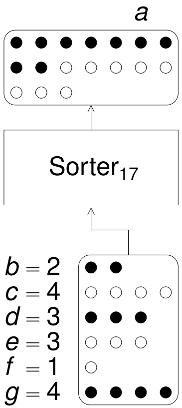
Outline

1. Primitives: Merging and Sorting Programs
2. Arithmetics Behind the Translation
3. Encoding the Summation
4. Enhancements
5. Experiments
6. Conclusions

1. Primitives: Merging and Sorting Programs

- We illustrate normalization designs using circuits
- Merging and sorting circuits have normal rule encodings
- Weight rules can be normalized using these primitives





2. Arithmetics Behind the Translation

- Suppose we have a weight rule of the form

$$\begin{aligned} a \leftarrow 31 \leqslant \langle b = 13, c = 7, d = 1, e = 11, f = 19, \\ g = 19, h = 10, \mathrm{not}\, i = 13, \mathrm{not}\, j = 6, \\ \mathrm{not}\, k = 13, \mathrm{not}\, l = 3, \mathrm{not}\, m = 4\rangle \end{aligned}$$

- ... and an answer set $M = \{a, c, d, e, i, k, \ldots\}$

- Summing the weights of satisfied body literals gives

$$7 + 1 + 11 + 6 + 3 + 4 = 32$$

- Question: How to do this with circuits?

Summing in Mixed-Radix Bases

- Using the mixed-radix base $B = 3, 2, \infty$:

	6	3	1
$c = 7$	•		•
$d = 1$			•
$e = 11$	•	•	••
not $j = 6$	•		
not $l = 3$		•	
not $m = 4$		•	•
$\Sigma = 32$	•••	•••	•••••
$\Sigma = 32$	•••	••••	••
$\Sigma = 32$	•••••		••
$bound = 31$	•••••		•

- Eén and Sörensson, JSAT'06

Simplifying Bound Checking with Tares

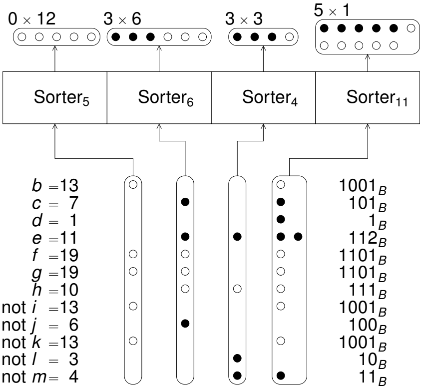
- Using the mixed-radix base $B = 3, 2, \infty$ and tare $t = 5$:

	6	3	1
$\Sigma = 32$	●●●	●●●	●●●●●
$t = \;\; 5$		●	●●
$\Sigma + t = 37$	●●●	●●●●	●●●●●●●
$\Sigma + t = 37$	●●●	●●●●●●	$\ast$
$\Sigma + t = 37$	●●●●●●	$\ast$	$\ast$
$bound + t = 36$	●●●●●●		

- Lexicographical comparison becomes trivial
- It suffices to know the most significant digit of the sum
- Bailleux, Boufkhad, and Roussel, SAT'09

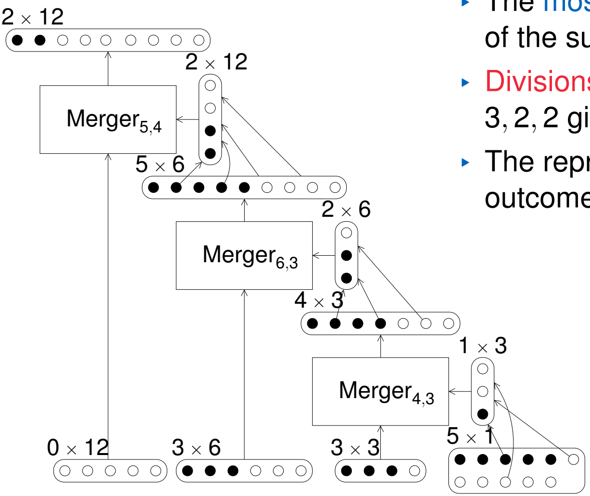
Digit-wise Summing

Normalization of $a \leftarrow 31 \leqslant [b = 13, c = 7, \ldots, \text{not } m = 4]$



Base $B = 3, 2, 2, \infty$ and answer set $M = \{a, c, d, e, i, k, \ldots\}$

Carry Propagation



- The most significant digit of the sum is computed
- Divisions by base radices 3, 2, 2 give carries
- The representation of the outcome becomes unique

4. Enhancements

- Several aspects of the translation can be adjusted
- Choices can be made between
  - types of mergers
  - mixed-radix bases
  - input arrangement in merge-sorting
- These choices affect translation size directly and through impacts on shared structure

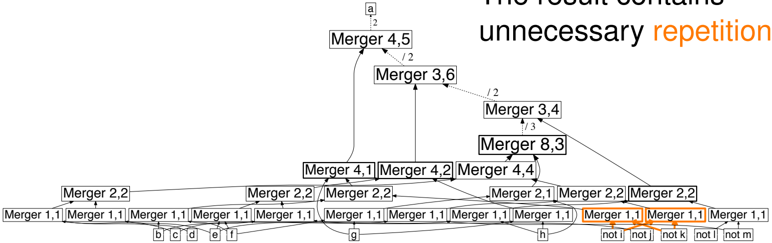
Mixed-Radix Base Selection

- Eén and Sörensson, JSAT'06
  - Enumerating bases consiting of primes < 20
- Bailleux, Boufkhad, and Roussel, SAT'09
  - Using binary bases
- Codish, Fekete, Fuhs, and Schneider-Kamp, TACAS'11
  - Searching optimal bases with sophisticated algorithms
- Our approach:
  - Radices are selected from least to most significant
  - Prime numbers are considered as candidates
  - Effects on translation size are heuristically estimated
  - The most promising prime is chosen

repeat

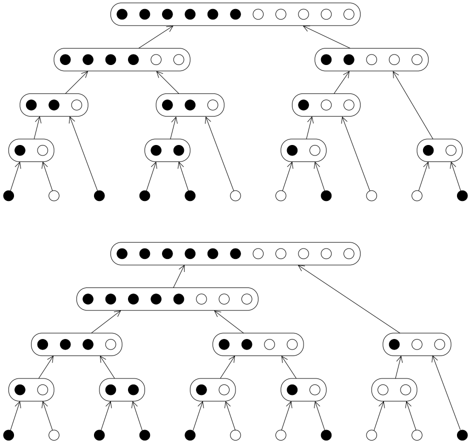
Implementation without Structure Sharing

- Normalization of $a \leftarrow 31 \leqslant [b = 13, c = 7, \ldots, \text{not}\, m = 4]$



- Sorters are implemented via merge-sorting
- The result contains unnecessary repetition

Restructuring Merge-Sorters

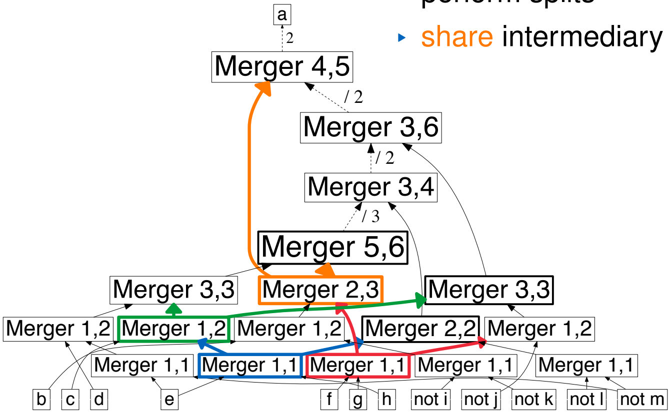


- Input can be arranged and divided freely
- Different choices lead to different structure
- With the right choices, shared input between sorters leads to common structure

Structure Sharing Result

We use a greedy algorithm to:

- perform splits
- share intermediary results



5. Experiments

- The translation is implemented in LP2NORMAL2 with configurable choices of bases and sharing
- For selected benchmarks, the proposed translation improves on the runtime of CLASP

		Mixed		Binary		
Benchmark	Native	Shared	Independent	Shared	Independent	SWC
Bayes-Find	202	**30**	164	246	165	1,721
Bayes-Prove	1,391	**492**	1,316	631	890	2,587
Markov-Find	2,426	2,770	**1,845**	2,682	2,966	5,224
Markov-Prove	**2,251**	3,294	3,428	3,255	3,229	5,402
Fastfood	**10,277**	12,843	14,156	13,756	13,479	17,867
Inc-Scheduling	**257**	1,340	1,330	1,481	1,581	
Nomystery	4,907	4,236	**3,332**	4,290	3,512	4,739
Summary	**21,715**	25,009	25,576	26,345	25,827	

6. Conclusions

We propose new ways to normalize weight rules, incorporating:

- Mixed-radix bases for concise representation of weights
- Tares for simplified bound checking
- Efficient primitives for digit-wise operations

Contributions:

- Structure sharing algorithm
- Base selection heuristic
- Generalization of cardinality translations for weight rules
- Selective and automated configuration of mergers