

# Propagation of ESCL Cardinality Constraints with Respect to CEP Queries

Thanh Son Dang

Bachelor thesis

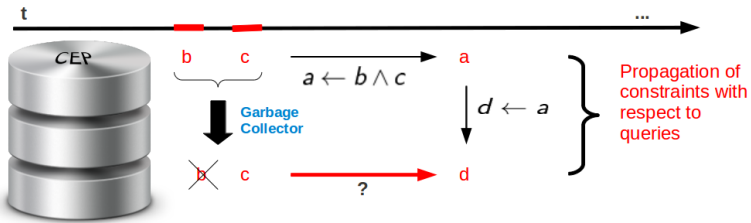
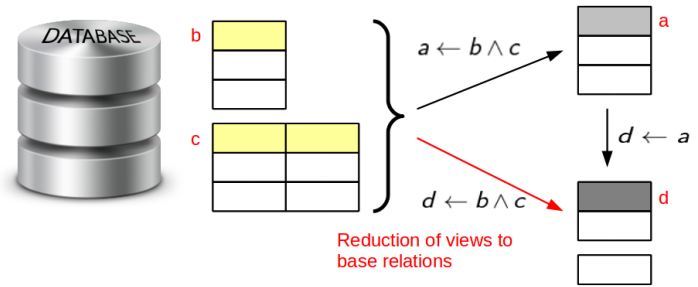
Supervisors: François Bry, Olga Poppe

Institute for Informatics, LMU Munich, Germany

December 15, 2011

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work



**Goal:** Reduce the number of manually specified constraints

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

ESCL cardinality constraint:  $i : \exists_c a \leftarrow$

where  $c$  is cardinality specification,  $i$  is time interval.

ESCL cardinality constraint:  $i : \exists_c a \leftarrow$

where  $c$  is cardinality specification,  $i$  is time interval.

- ▶ Simple Cardinality Specification:  $i : \exists_{\theta n} a \leftarrow$   
 where  $\theta \in \{=, \leq, \geq\}$ ,  $n \in \mathbb{N}_0$ ,  
 e.g.  $i : \exists_{=1} a \leftarrow$ ,  $i : \exists_{\geq 4} b \leftarrow$ ,  $i : \exists_{\leq 8} c \leftarrow$
- ▶ Complex Cardinality Specification: e.g.  $i : \exists_{\geq 4 \vee =1} a \leftarrow$ ,  
 $i : \exists_{=1 \vee (\geq 4 \wedge \leq 8)} \leftarrow$

$= 1$  corresponds  $[1, 1]$ ,  $\geq 4$  corresponds  $[4, \infty)$ .

$\geq 4 \wedge \leq 8$  corresponds to  $[4, 8] = [4, \infty) \cap [0, 8]$

$= 1 \vee (\geq 4 \wedge \leq 8)$  corresponds to  $[1, 1] \cup [4, 8]$

Simple cardinality specifications are always valid.

Invalid cardinality specification:  $i : \exists_{=1 \wedge \geq 4}$ , since  $[1, 1] \cap [4, \infty) = \emptyset$ .

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization**
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

- 1) *Recognition of query unsatisfiability*
- 2) *Query suspension*
- 3) *Scan reduction*
- 4) *Query termination*
- 5) *Choice of join algorithms*
- 6) *Join introduction*

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions**
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

- 1) All literals in a query body are positive
- 2) Constraints are facts
- 3) Constraints involve  $\leq, \geq, =$  (no  $<, >, \neq$ )
- 4) Atoms in the constraint heads are the same as the atoms in the query body.
- 5) Constraints are valid and simplified

**Example:**

*Query :*            *Range 1h :*  $h \leftarrow l_1 \vee (l_2 \wedge l_3)$

*Constraints :*    *Range 1h :*  $\exists_{=2 \vee \geq 4} l_1 \leftarrow$

*Range 3h :*  $\exists_{\leq 2} l_2 \leftarrow$

*Range 30min :*  $\exists_{\leq 4} l_3 \leftarrow$

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges**
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

- 1) Arbitrarily complex query
- 2) Arbitrarily complex cardinality specifications
- 3) The query evaluation time does not coincide with the validity time of the constraints

**Example:**

*Query* :  $Range\ 1h : h \leftarrow l_1 \vee (l_2 \wedge l_3)$

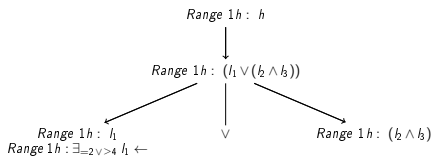
*Constraints* :  $Range\ 1h : \exists_{=2 \vee \geq 4} l_1 \leftarrow$

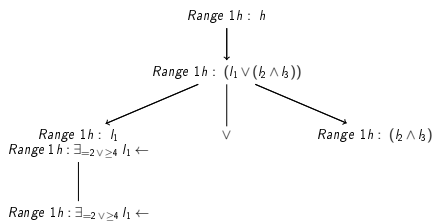
$Range\ 3h : \exists_{\leq 2} l_2 \leftarrow$

$Range\ 30min : \exists_{\leq 4} l_3 \leftarrow$

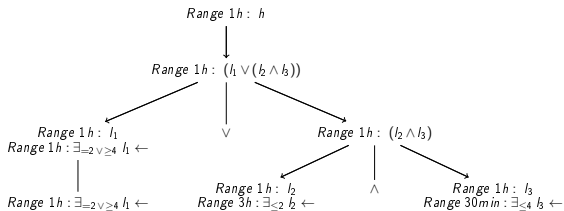
**Goal:**  $Range\ 1h : \exists_? h \leftarrow$

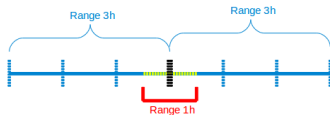
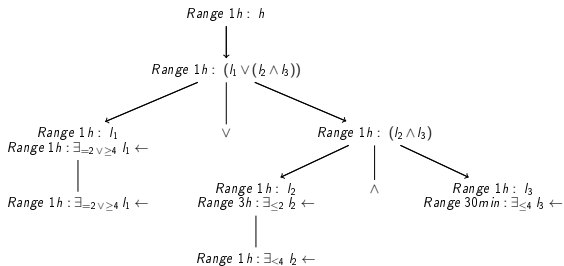
$$\begin{array}{c} \text{Range } 1h : h \\ \downarrow \\ \text{Range } 1h : (h_1 \vee (h_2 \wedge h_3)) \end{array}$$

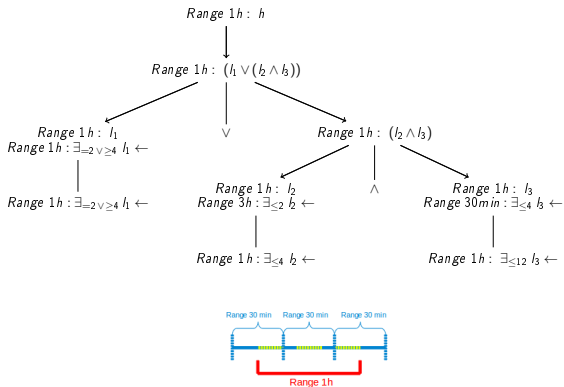




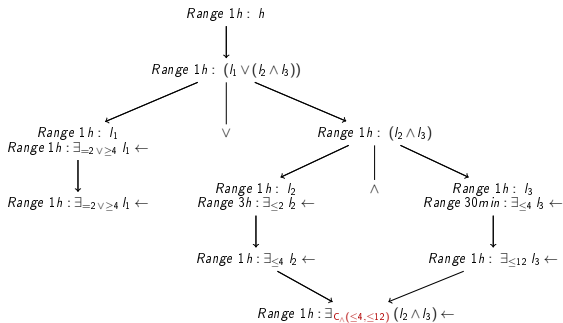
# Example





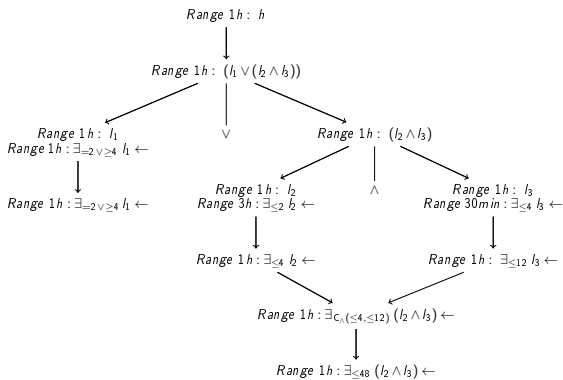


# Example

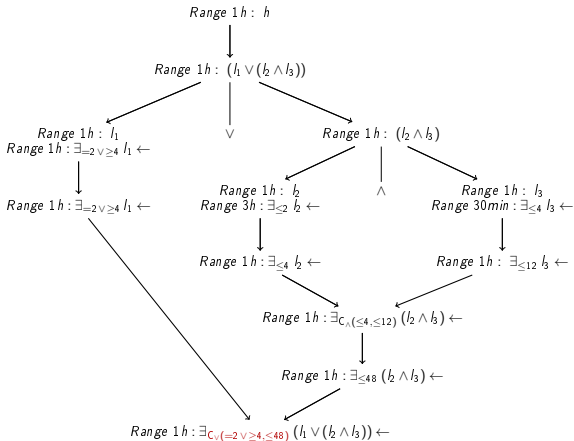


$C_B \backslash C_A$	= 0	= c	$\leq a$	$\geq a$
= 0	= 0	= 0	= 0	= 0
= d	= 0	= cd	$\leq ad$	$\geq ad$
$\leq b$	= 0	$\leq cb$	$\leq ab$	$= 0 \vee \geq a$
$\geq b$	= 0	$\geq cb$	$= 0 \vee \geq b$	$\geq ab$

$$C_A(\leq 4, \leq 12) = \leq 4 \cdot 12 = \leq 48$$

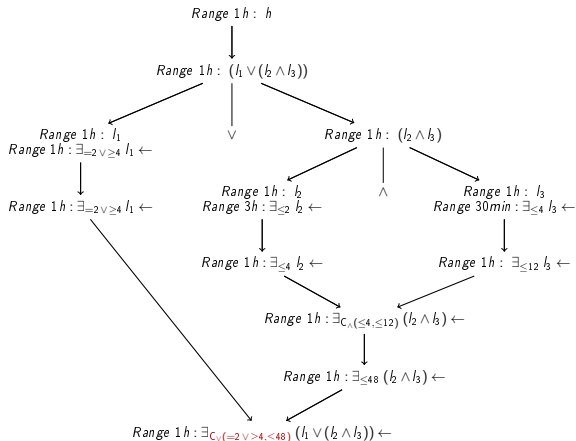


# Example



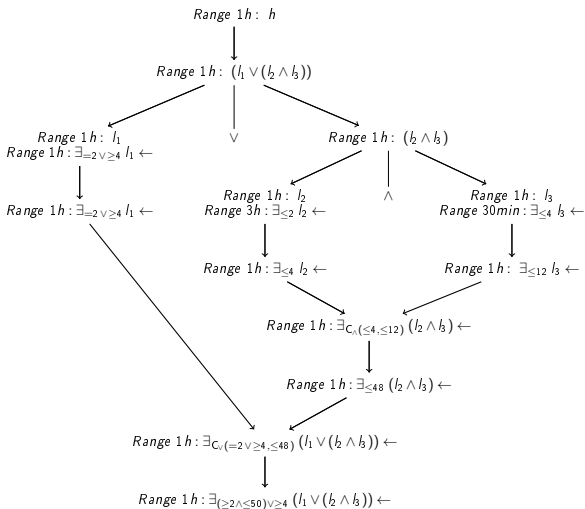
$C_B \backslash C_A$	$C_{A_1} \vee C_{A_2}$	$C_{A_1} \wedge C_{A_2}$
$C_{B_1} \vee C_{B_2}$	$C_V(C_{A_1}, C_{B_1}) \vee C_V(C_{A_2}, C_{B_1}) \vee C_V(C_{A_1}, C_{B_2}) \vee C_V(C_{A_2}, C_{B_2})$	$(C_V(C_{A_1}, C_{B_1}) \wedge C_V(C_{A_2}, C_{B_1})) \vee (C_V(C_{A_1}, C_{B_2}) \wedge C_V(C_{A_2}, C_{B_2}))$
$C_{B_1} \wedge C_{B_2}$	$(C_V(C_{A_1}, C_{B_1}) \wedge C_V(C_{A_1}, C_{B_2})) \vee (C_V(C_{A_2}, C_{B_1}) \wedge C_V(C_{A_2}, C_{B_2}))$	$C_V(C_{A_1}, C_{B_1}) \wedge C_V(C_{A_2}, C_{B_1}) \wedge C_V(C_{A_1}, C_{B_2}) \wedge C_V(C_{A_2}, C_{B_2})$

$$C_V(=2v \geq 4, \leq 48) = C_V(=2, \leq 48) \vee C_V(\geq 4, \leq 48)$$

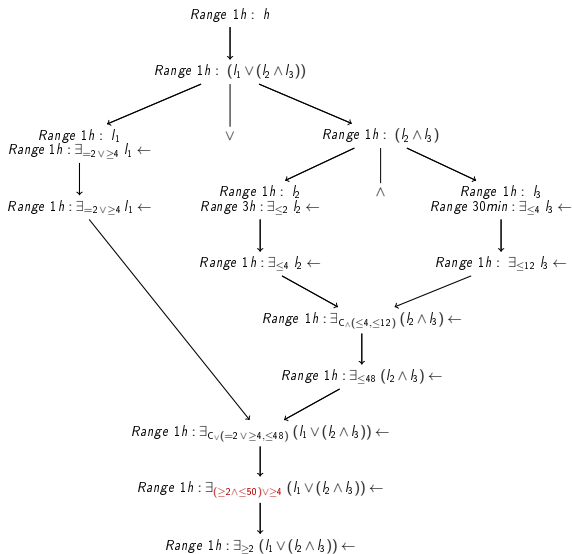


$C_B \backslash C_A$	$= a$	$\leq a$	$\geq a$
$= b$	$= a + b$	$\geq b \wedge \leq a + b$	$\geq a + b$
$\leq b$	$\geq a \wedge \leq a + b$	$\leq a + b$	$\geq a$
$\geq b$	$\geq a + b$	$\geq b$	$\geq a + b$

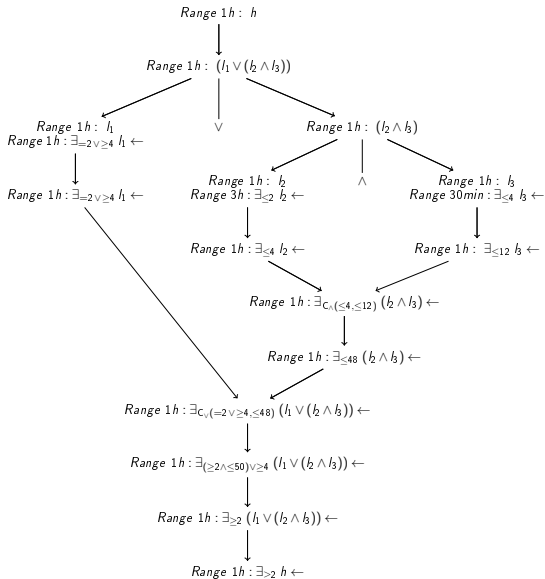
$$C_V(= 2v \geq 4, \leq 48) = C_V(= 2, \leq 48) \vee C_V(\geq 4, \leq 48) \\ = (\geq 2 \wedge \leq 50) \vee (\geq 4)$$

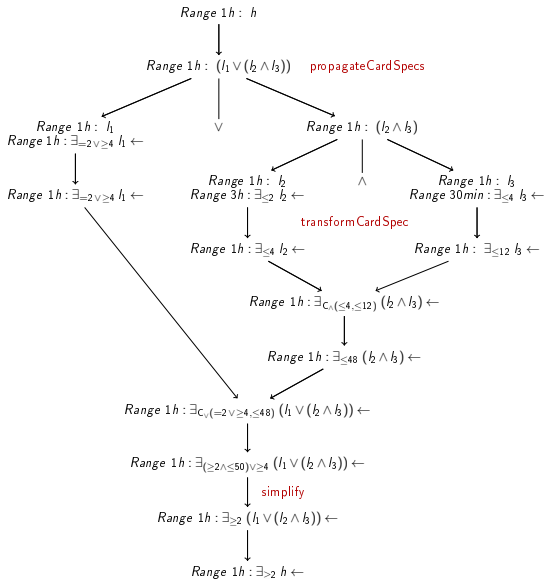


# Example



$(\geq 2 \wedge \leq 50) \vee \geq 4$  corresponds  $[2, 50] \cup [4, \infty) = [2, \infty)$





- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms**
- 7 Results
- 8 Future Work

	Termination	Complexity	Correctness	Completeness
simplify	✓	$O(n)$	✓	✓
transform-CardSpec	✓	$O(n^2)$	✓	✓
propagate-CardSpec	✓	if $m = 1$ : $O(n^2)$  if $m \geq 2$ : $O(2^{4m-6} \cdot n^{2m-2})$	✓	✓

where  $m$  is the number of the atoms in the query body, and  $n$  denotes the maximal number of simple cardinality specifications of the cardinality constraints for the atoms.

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

Algorithm propagating of the ESCL cardinality constraints with respect to StreamLog queries such that

- 1) The query and constraints may be arbitrary complex
- 2) The query evaluation time does not coincide with the validity time of the constraints

Proofs of the termination, correctness and completeness of the algorithm, investigation of its complexity

- 1 Introduction and Motivation
- 2 Classification of ESCL Cardinality Constraints
- 3 Use of Cardinality Constraints for Semantic Optimization
- 4 Assumptions
- 5 Challenges
- 6 Properties of the algorithms
- 7 Results
- 8 Future Work

- 1) Propagation of ESCL cardinality constraints independently from the assumptions
- 2) Propagation of ESCL causal, temporal, and data constraints with respect to StreamLog queries
- 3) Derivation of ESCL constraints from StreamLog queries independently from other ESCL constraints
- 4) Implementation and experimental evaluation of the whole approach

Thank you for your kind attention!