Logical Rules for a Lexicographic Order Constraint Solver

Thom Frühwirth

Faculty of Computer Science
University of Ulm, Germany
www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/

CHR 2005, Barcelona, October 2005
Lexicographic Order

Many Applications

- Comparing and sorting, e.g. strings
- Termination analysis, e.g. rewrite systems
- Symmetry breaking in constraint programming
- Modelling preferences in constraint programming

Different algorithms for the lex constraint

- Non-incremental pointer-based imperative pseudo code, 45 lines
  Frisch/Hnich/Kiziltan et. al. CP 2002.
- Finite automaton based on 7 cases, 42 lines
- Simpler automata, interpreted, 16 lines
  Beldiceanu/Carlsson/Petit, CP 2004.

Non-intuitive non-logical code. Hard to analyse.
Constraint Handling Rules (CHR)

- **CHR**: Constraint programming language for Computational Logic
- Multi-headed guarded committed-choice rules transform multi-set of constraints until exhaustion
- Ideal for executable specifications and rapid prototyping
- **Efficient**: all algorithms implementable with optimal time complexity
- Incrementality and concurrency for free (on-line, any-time)
- Logical and operational semantics coincide strongly
- High-level supports semi-automatic program analysis + transformation:
  Confluence + completion, termination + time complexity, correctness...
- Implementations in most Prolog systems, Java, Haskell
- 100s of applications from types, time tabling to cancer diagnosis

Use CHR for a global lexic constraint?

Thom Frühwirth
Logical Rules for a Lexicographic Order Constraint Solver
Lexicographic Order

Definition
Given two sequences \( l_1 \) and \( l_2 \) of length \( n \), \([x_1, \ldots, x_n]\) and \([y_1, \ldots, y_n]\), then \( l_1 \preceq_{\text{lex}} l_2 \) iff \( n=0 \) or \( x_1 < y_1 \) or \( x_1 = y_1 \) and \([x_2, \ldots, x_n]\) \( \preceq_{\text{lex}} [y_2, \ldots, y_n] \).

Logical Specification
\[
l_1 \preceq_{\text{lex}} l_2 \iff \begin{cases} (l_1 = \[] \land l_2 = \[]) \lor \\ (l_1 = [x | l_1'] \land l_2 = [y | l_2'] \land x < y) \lor \\ (l_1 = [x | l_1'] \land l_2 = [y | l_2'] \land x = y \land l_1' \preceq_{\text{lex}} l_2')
\end{cases}
\]

CHR Implementation
\[
l_1 \, @ \, [[] \, \text{lex} \, [] \, \Rightarrow \, \text{true}. \\
l_2 \, @ \, [X|L1] \, \text{lex} \, [Y|L2] \, \Rightarrow \, X < Y \mid \text{true}. \\
l_3 \, @ \, [X|L1] \, \text{lex} \, [Y|L2] \, \Rightarrow \, X = Y \mid L1 \, \text{lex} \, L2.
\]
Lexicographic Order

Definition
Given two sequences \( l_1 \) and \( l_2 \) of length \( n \), \([x_1, \ldots, x_n]\) and \([y_1, \ldots, y_n]\), then \( l_1 \leq_{\text{lex}} l_2 \) iff \( n=0 \) or \( x_1 < y_1 \) or \( x_1 = y_1 \) and \([x_2, \ldots, x_n] \leq_{\text{lex}} [y_2, \ldots, y_n] \).

Logical Specification
\[
l_1 \leq_{\text{lex}} l_2 \iff \left( l_1=[\] \land l_2=[\] \right) \lor \\
\left( l_1=[x | l'_1] \land l_2=[y | l'_2] \land x < y \right) \lor \\
\left( l_1=[x | l'_1] \land l_2=[y | l'_2] \land x = y \land l'_1 \leq_{\text{lex}} l'_2 \right)
\]

CHR Implementation
\[
l_1 \odot [] \text{ lex } [] \iff \text{ true.}
\]
\[
l_2 \odot [X | L1] \text{ lex } [Y | L2] \iff X < Y \mid \text{ true.}
\]
\[
l_3 \odot [X | L1] \text{ lex } [Y | L2] \iff X = Y \mid L_1 \text{ lex } L_2.
\]
Lexicographic Order

Definition
Given two sequences $l_1$ and $l_2$ of length $n$, $[x_1, \ldots, x_n]$ and $[y_1, \ldots, y_n]$, then $l_1 \leq_{\text{lex}} l_2$ iff $n=0$ or $x_1 < y_1$ or $x_1 = y_1$ and $[x_2, \ldots, x_n] \leq_{\text{lex}} [y_2, \ldots, y_n]$.

Logical Specification
$l_1 \leq_{\text{lex}} l_2 \iff (l_1 = [] \land l_2 = []) \lor (l_1 = [x|l'_1] \land l_2 = [y|l'_2] \land x < y) \lor (l_1 = [x|l'_1] \land l_2 = [y|l'_2] \land x = y \land l'_1 \leq_{\text{lex}} l'_2)$

CHR Implementation

\begin{align*}
l_1 & @ [] \text{ lex } [] \iff \text{ true.} \\
l_2 & @ [X|L1] \text{ lex } [Y|L2] \iff X < Y \mid \text{ true.} \\
l_3 & @ [X|L1] \text{ lex } [Y|L2] \iff X = Y \mid L1 \text{ lex } L2.
\end{align*}
First Implementation in CHR

\[ l1 \circ [] \text{ lex } [] \iff \text{true}. \]
\[ l2 \circ [X|L1] \text{ lex } [Y|L2] \iff X<Y \mid \text{true}. \]
\[ l3 \circ [X|L1] \text{ lex } [Y|L2] \iff X=Y \mid L1 \text{ lex } L2. \]

Queries

\[ [1] \text{ lex } [2] \rightarrow_{12} \text{true}. \]
\[ [X] \text{ lex } [X] \rightarrow_{13} [] \text{ lex } [] \rightarrow_{11} \text{true}. \]
\[ [X] \text{ lex } [Y], X<Y \rightarrow_{12} X<Y. \]
\[ [X] \text{ lex } [Y], X\neq Y. \]
\[ [X] \text{ lex } [Y], X>Y. \]
\[ [X] \text{ lex } [Y], X\neq Y. \]

No propagation yet :-(
First Implementation in CHR

l1 @ [] lex [] <=> true.
l2 @ [X|L1] lex [Y|L2] <=> X<Y | true.
l3 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.

Queries
[X] lex [X] →_{l3} [] lex [] →_{l1} true.
[X] lex [Y], X<Y →_{l2} X<Y.
[X] lex [Y].
[X] lex [Y], X>Y.
[X] lex [Y], X<>Y.
No propagation yet :-(
First Implementation in CHR

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.

Queries
[X] lex [X] \rightarrow_{13} [] lex [] \rightarrow_{11} true.
[X] lex [Y], X<Y \rightarrow_{12} X<Y.
[X] lex [Y].
[X] lex [Y], X>Y.
[X] lex [Y], X<>Y.
No propagation yet :-(

Thom Frühwirth  Logical Rules for a Lexicographic Order Constraint Solver
First Implementation in CHR

l1 @ [] lex [] <=> true.
l2 @ [X|L1] lex [Y|L2] <=> X<Y | true.
l3 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.

Queries

[X] lex [X] →13 [] lex [] →11 true.
[X] lex [Y], X<Y →12 X<Y.
[X] lex [Y].
[X] lex [Y], X>Y.
[X] lex [Y], X<>Y.
No propagation yet :-(

Thom Frühwirth
Logical Rules for a Lexicographic Order Constraint Solver
First Implementation in CHR

\begin{align*}
11 & @ [] \text{ lex } [] \iff \text{true.} \\
12 & @ [X|L1] \text{ lex } [Y|L2] \iff X<Y \ | \text{true.} \\
13 & @ [X|L1] \text{ lex } [Y|L2] \iff X=Y \ | \text{L1 lex L2.}
\end{align*}

Queries
\begin{align*}
[1] & \text{ lex } [2] \rightarrow_{12} \text{true.} \\
[X] & \text{ lex } [X] \rightarrow_{13} [] \text{ lex } [] \rightarrow_{11} \text{true.} \\
[X] & \text{ lex } [Y], \ X<Y \rightarrow_{12} X<Y. \\
[X] & \text{ lex } [Y]. \\
[X] & \text{ lex } [Y], \ X>Y. \\
[X] & \text{ lex } [Y], \ X<>Y. \\
\text{No propagation yet :-(}
\end{align*}
First Implementation in CHR

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.

Queries

[X] lex [X] \rightarrow_{13} [] lex [] \rightarrow_{11} true.
[X] lex [Y], X<Y \rightarrow_{12} X<Y.
[X] lex [Y].
[X] lex [Y], X>Y.
[X] lex [Y], X<>Y.

No propagation yet :-(
Adding Propagation

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.

Queries

[X] lex [Y] ->14 [X] lex [Y], X=<Y.
[X] lex [Y], X<>Y ->14 [X] lex [Y], X<Y ->12 X<Y.

[R1,R2,R3] lex [T1,T2,T3], R2=T2, R3>T3 doesn’t add R1<T1 :-(

Thom Frühwirth
Logical Rules for a Lexicographic Order Constraint Solver
Adding Propagation

l1 @ [] lex [] <=> true.
l2 @ [X|L1] lex [Y|L2] <=> X<Y | true.
l3 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
l4 @ [X|L1] lex [Y|L2] ==> X=<Y.

Queries

[X] lex [Y] →_{14} [X] lex [Y], X=<Y.
[X] lex [Y], X>Y →_{14} [X] lex [Y], X>Y, X=<Y →_{built-in} fail.
[X] lex [Y], X<>Y →_{14} [X] lex [Y], X<Y →_{12} X<Y.

[R1,R2,R3] lex [T1,T2,T3], R2=T2, R3>T3 doesn’t add R1<T1 :-(
Adding Propagation

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.

Queries

[X] lex [Y] →14 [X] lex [Y], X=<Y.
[X] lex [Y], X>Y →14 [X] lex [Y], X>Y, X=<Y →built−in fail.
[X] lex [Y], X<>Y →14 [X] lex [Y], X<Y →12 X<Y.

[R1,R2,R3] lex [T1,T2,T3], R2=T2, R3>T3 doesn’t add R1<T1 :-(

Thom Frühwirth  Logical Rules for a Lexicographic Order Constraint Solver
Adding Propagation

\[ l_{11} @ [] \text{ lex } [] \leftrightarrow \text{ true.} \]
\[ l_{12} @ [X|L1] \text{ lex } [Y|L2] \leftrightarrow X<Y \mid \text{ true.} \]
\[ l_{13} @ [X|L1] \text{ lex } [Y|L2] \leftrightarrow X=Y \mid L1 \text{ lex } L2. \]
\[ l_{14} @ [X|L1] \text{ lex } [Y|L2] \rightarrow X=<Y. \]

Queries
\[ [X] \text{ lex } [Y] \rightarrow_{14} [X] \text{ lex } [Y], X=<Y. \]
\[ [X] \text{ lex } [Y], X>Y \rightarrow_{14} [X] \text{ lex } [Y], X>Y, X=<Y \rightarrow \text{ built-in fail.} \]
\[ [X] \text{ lex } [Y], X<>Y \rightarrow_{14} [X] \text{ lex } [Y], X<Y \rightarrow_{12} X<Y. \]

\[ [R1,R2,R3] \text{ lex } [T1,T2,T3], R2=T2, R3>T3 \text{ doesn't add } R1<T1 \text{ :-}( \]

Thom Frühwirth  Logical Rules for a Lexicographic Order Constraint Solver
Adding Simplification

\[11 \odot \emptyset \text{ lex } \emptyset \iff \text{ true.}\]
\[12 \odot [X|L1] \text{ lex } [Y|L2] \iff X<Y \ | \text{ true.}\]
\[13 \odot [X|L1] \text{ lex } [Y|L2] \iff X=Y \ | \ L1 \text{ lex } L2.\]
\[14 \odot [X|L1] \text{ lex } [Y|L2] \implies X=\leq Y.\]
\[15 \odot [X,U|L1] \text{ lex } [Y,V|L2] \iff U>V \ | \ X<Y.\]
\[16 \odot [X,U|L1] \text{ lex } [Y,V|L2] \iff U=V \ | \ [X|L1] \text{ lex } [Y|L2].\]

\[[R1,R2,R3] \text{ lex } [T1,T2,T3], \ R2=\leq T2, \ R3>\leq T3\]
\[
\to_{16} R2=\leq T2, \ R3>\leq T3, \ [R1,R3] \text{ lex } [T1,T3]\]
\[
\to_{15} R2=\leq T2, \ R3>\leq T3, \ R1<\leq T1.\]

\[\text{[] R1,R2,R3] lex [T1,T2,T3], R2>=T2, R3>T3 doesn’t add R1<T1 :-(}\]
Adding Simplification

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.
16 @ [X,U|L1] lex [Y,V|L2] <=> U=V | [X|L1] lex [Y|L2].

[R1,R2,R3] lex [T1,T2,T3], R2=T2, R3>T3
→16 R2=T2, R3>T3, [R1,R3] lex [T1,T3]
→15 R2=T2, R3>T3, R1<T1.

[R1,R2,R3] lex [T1,T2,T3], R2>=T2, R3>T3 doesn’t add R1<T1 :-(
Adding Simplification

$11 \odot \emptyset \text{ lex } \emptyset \iff \text{ true.}$

$12 \odot [X|L1] \text{ lex } [Y|L2] \iff X \lt Y \mid \text{ true.}$

$13 \odot [X|L1] \text{ lex } [Y|L2] \iff X = Y \mid L1 \text{ lex } L2.$

$14 \odot [X|L1] \text{ lex } [Y|L2] \implies X \leq Y.$

$15 \odot [X,U|L1] \text{ lex } [Y,V|L2] \iff U \gt V \mid X \lt Y.$

$16 \odot [X,U|L1] \text{ lex } [Y,V|L2] \iff U = V \mid [X|L1] \text{ lex } [Y|L2].$

$[R1,R2,R3] \text{ lex } [T1,T2,T3], \ R2 = T2, \ R3 \gt T3$

$\rightarrow_{16} \ R2 = T2, \ R3 \gt T3, \ [R1,R3] \text{ lex } [T1,T3]$

$\rightarrow_{15} \ R2 = T2, \ R3 \gt T3, \ R1 \lt T1.$

$[R1,R2,R3] \text{ lex } [T1,T2,T3], \ R2 \geq T2, \ R3 \gt T3 \text{ doesn't add } R1 \lt T1 :($
Turning Simplification into Propagation

\[ l_{11} @ [] \text{lex} [] \Leftrightarrow \text{true.} \]
\[ l_{12} @ [X|L_1] \text{lex} [Y|L_2] \Leftrightarrow X < Y \mid \text{true.} \]
\[ l_{13} @ [X|L_1] \text{lex} [Y|L_2] \Leftrightarrow X = Y \mid L_1 \text{lex} L_2. \]
\[ l_{14} @ [X|L_1] \text{lex} [Y|L_2] \implies X \leq Y. \]
\[ l_{15} @ [X,U|L_1] \text{lex} [Y,V|L_2] \Leftrightarrow U > V \mid X < Y. \]
\[ l_{16}' @ [X,U|L_1] \text{lex} [Y,V|L_2] \implies U = V \mid [X|L_1] \text{lex} [Y|L_2]. \]

\[ [R_1,R_2,R_3] \text{lex} [T_1,T_2,T_3], R_2 \geq T_2, R_3 > T_3 \rightarrow_{16}' \]
\[ [R_1,R_2,R_3] \text{lex} [T_1,T_2,T_3], R_2 \geq T_2, R_3 > T_3, [R_1,R_3] \text{lex} [T_1,T_3] \rightarrow_{15} [R_1,R_2,R_3] \text{lex} [T_1,T_2,T_3], R_2 \geq T_2, R_3 > T_3, R_1 < T_1 \rightarrow_{12} R_2 = T_2, R_3 > T_3, R_1 < T_1. \]

Quadratic worst-case time complexity, if after complete propagation with \[ l_{16}' \] \( O(n^2) \) variable pairs are removed one by one by rule \( l_{13} \).
Turning Simplification into Propagation

\begin{align*}
l_1 & @ [] \text{ lex } [] \iff \text{ true.} \\
l_2 & @ [X|L_1] \text{ lex } [Y|L_2] \iff X < Y \mid \text{ true.} \\
l_3 & @ [X|L_1] \text{ lex } [Y|L_2] \iff X = Y \mid L_1 \text{ lex } L_2. \\
l_4 & @ [X|L_1] \text{ lex } [Y|L_2] \implies X \leq Y. \\
l_5 & @ [X,U|L_1] \text{ lex } [Y,V|L_2] \iff U > V \mid X < Y. \\
l_6' & @ [X,U|L_1] \text{ lex } [Y,V|L_2] \implies U \geq V \mid [X|L_1] \text{ lex } [Y|L_2].
\end{align*}

\begin{align*}
[R_1,R_2,R_3] & \text{ lex } [T_1,T_2,T_3], \ R_2 \geq T_2, R_3 > T_3 \rightarrow_{16'} \\
[R_1,R_2,R_3] & \text{ lex } [T_1,T_2,T_3], \ R_2 \geq T_2, R_3 > T_3, \ [R_1,R_3] \text{ lex } [T_1,T_3] \\
\rightarrow_{15} [R_1,R_2,R_3] & \text{ lex } [T_1,T_2,T_3], \ R_2 \geq T_2, R_3 > T_3, \ R_1 < T_1 \\
\rightarrow_{12} [R_2 = T_2] & , \ R_3 > T_3, \ R_1 < T_1.
\end{align*}

Quadratic worst-case time complexity, if after complete propagation with \(16'\) \(O(n^2)\) variable pairs are removed one by one by rule \(13\).
Turning Simplification into Propagation

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.
16' @ [X,U|L1] lex [Y,V|L2] ==> U>=V | [X|L1] lex [Y|L2].

[R1,R2,R3] lex [T1,T2,T3], R2>=T2,R3>T3 →_{16'}
[R1,R2,R3] lex [T1,T2,T3], R2>=T2,R3>T3, [R1,R3] lex [T1,T3] →_{15}
[R1,R2,R3] lex [T1,T2,T3], R2>=T2,R3>T3, R1<T1 →_{12} R2>=T2, R3>T3, R1<T1.

Quadratic worst-case time complexity, if after complete propagation with 16' $O(n^2)$ variable pairs are removed one by one by rule 13.
Improving Time Complexity

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.

The culprit propagation rule...

16' @ [X,U|L1] lex [Y,V|L2] ==> U>=V | [X|L1] lex [Y|L2].
Improving Time Complexity

```
11 @ [] lex [] <=true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] => X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.
```

...is turned into a logically equivalent simplification rule

```
16' @ [X,U|L1] lex [Y,V|L2] <=> U=V | [X|L1] lex [Y|L2],
[X,U|L1] lex [Y,V|L2].
```

but it does not terminate.
Improving Time Complexity

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X < Y | true.
13 @ [X|L1] lex [Y|L2] <=> X = Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X <= Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U > V | X < Y.

...this rule terminates, it is correct and efficient.

16’’ @ [X,U|L1] lex [Y,V|L2] <=> U >= V, L1 = [ ] | [X|L1] lex [Y|L2],
    [X,U] lex [Y,V].
Worst-Case Time Complexity

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.

Measure independent of constraint system (with its own complexity):
Number of atomic built-in constraints that are checked and imposed.
Proportional to the number of rule applications.
Worst-Case Time Complexity

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.

Number of rule applications depends on list length \( n \) of \( \text{lex} \) constraint.

- Propagation rule 14 applied at most once.
- Rules 11, 12 and 15 contribute constant number.
- Recursive rule 13: linear in list length.
- Recursive rule 16’’: linear, as \([X,U] \text{ lex } [Y,V] \) constant number.

Linear in list length \( n \). At most \( O(n) \) built-in constraints processed.
Confluence

Result of a query is always the same, no matter which of the applicable rules are applied.

\[
\begin{align*}
A & \mapsto B \\
A & \mapsto C \\
\hline
B & \mapsto^* D \\
C & \mapsto^* D
\end{align*}
\]

⇒ Independence from the order in which constraints processed.
⇒ Consistency of logical reading of the program.

Decidable, sufficient and necessary condition for confluence of terminating CHR programs through joinability of critical pairs.

Critical pair results from applying two rules to an overlap.
Overlap takes both rule heads and guards, shares some CHR constraints.

Confluence checker available.
Example for `lex`

13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.

Critical Pair from Overlap

```
[X,U|L1] lex [Y,V|L2],
X=Y,
U>V
```

```
X=Y, U>V, [U|L1] lex [V|L2]
14, built-in
fail
```

```
X=Y, U>V, X < Y
15
```

```
[13]←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←→
Example for \textit{lex}

\begin{align*}
13 \odot [X|L1] \text{ lex } [Y|L2] & \iff X=Y \mid L1 \text{ lex } L2. \\
15 \odot [X,U|L1] \text{ lex } [Y,V|L2] & \iff U>V \mid X<Y.
\end{align*}

\textbf{Critical Pair from Overlap}

\begin{align*}
[X,U|L1] \text{ lex } [Y,V|L2], & \quad X=Y, \quad U>V \\
X=Y, U>V, [U|L1] \text{ lex } [V|L2] & \iff X=Y, U>V, X < Y \\
14, \text{ built-in} & \iff \text{fail} \\
\text{built-in} & \iff \text{fail}
\end{align*}
Logical Correctness

11 @ [] lex [] <=> true.
12 @ [X|L1] lex [Y|L2] <=> X<Y | true.
13 @ [X|L1] lex [Y|L2] <=> X=Y | L1 lex L2.
14 @ [X|L1] lex [Y|L2] ==> X=<Y.
15 @ [X,U|L1] lex [Y,V|L2] <=> U>V | X<Y.
16’’@[X,U|L1] lex [Y,V|L2] <=> U>=V, L1=[_ _] | ...

Logical Reading

(\[
\begin{align*}
X<Y & \rightarrow ([X|L1] \preceq_{lex} [Y|L2]) \\
X=Y & \rightarrow ([X|L1] \preceq_{lex} [Y|L2] \\
& \quad ([X|L1] \preceq_{lex} [Y|L2]) \\
U>V & \rightarrow ([X, U|L1] \preceq_{lex} [Y, V|L2] \\
(U \geq V \land L1=[_ _]) & \rightarrow ([X, U|L1] \preceq_{lex} [Y, V|L2] \\
& \quad ([X, U|L1] \preceq_{lex} [Y, V] \land [X|L1] \preceq_{lex} [Y|L2]))
\end{align*}
\]
)
Logical Correctness

Logical Specification

\[ l_1 \preceq_{\text{lex}} l_2 \iff (l_1 = \emptyset \land l_2 = \emptyset) \lor (l_1 = [x | l'_1] \land l_2 = [y | l'_2] \land x < y) \lor (l_1 = [x | l'_1] \land l_2 = [y | l'_2] \land x = y \land l'_1 \preceq_{\text{lex}} l'_2) \]

Logical Reading must be logical consequence of above specification

\[
\begin{align*}
X < Y & \rightarrow ([X | L1] \preceq_{\text{lex}} [Y | L2]) \quad \leftrightarrow L1 \preceq_{\text{lex}} L2) \\
X = Y & \rightarrow ([X | L1] \preceq_{\text{lex}} [Y | L2] \quad \rightarrow X \leq Y) \\
U > V & \rightarrow ([X, U | L1] \preceq_{\text{lex}} [Y, V | L2] \leftrightarrow X < Y) \\
(U \geq V \land L1 = [\_ | \_]) & \rightarrow ([X, U | L1] \preceq_{\text{lex}} [Y, V | L2] (\leftrightarrow ([X, U] \preceq_{\text{lex}} [Y, V] \land [X | L1] \preceq_{\text{lex}} [Y | L2]))
\end{align*}
\]
Completeness

Local Consistency: Compute all implied inequalities from lexicographic constraint and given inequality.

We already know that the solver is correct and confluent.

Correctness implies: can only propagate too little, not too much / wrong. Do we propagate enough?

Confluence means: if we find a way to propagate, then we will always propagate, no matter which rules are applied.
Completeness

The \texttt{lex} constraint admits $n + 1$ solutions of the form

$$x_1 = y_1 \land x_2 = y_2 \land \ldots \land x_{i-1} = y_{i-1} \land x_i < y_i \quad (1 \leq i \leq n+1),$$

$x_i < y_i$ is dropped if $i = n+1$.

We write this as $(=)^{i-1} [\prec]$. $i$ is the position of the solution.

Inequality constraints can be added to a given \texttt{lex} constraint so that any subset of solutions is possible.

We want to propagate inequality constraints from the disjunction of these solutions to a given \texttt{lex} constraint.
Completeness

- **Forward Propagation**
  From the disjunction we must propagate such that \((\equiv)^{j-1}[\leq]\), where \(j\) is the smallest position of any solution.

  We can find \(j\) by looking for the smallest position that admits \(<\).
  A position that does not admit a solution must be \(>\) or \(\geq\).
  After \(>\) there can be no more solution.
  \(\equiv\) positions can be ignored.

  There is no solution if \((\equiv)^* >\).

  \(\Rightarrow\) Propagation rule 14 imposes \(\leq\) on any current first position and the recursive simplification rule 13 removes leading \(\equiv\).
Completeness

• **Backward Propagation**

If there is only one solution, we must strengthen propagation such that $(\leq)^{j-1}[<]$.

⇒ Rule 15: $>$ holds for the second position, so $<$ must hold for the first position. Recursive rule 16’’ watches all $\geq$ constraints until $>$. 
Conclusions

Lexicographic Order Constraint Solver in CHR

- Executable specification: **short, concise**
  using recursive decomposition and propagation
- **Incremental** and concurrent: by nature of CHR
- **Independent** of underlying constraint system
- **Complete**: propagates as much as possible
- **Efficient**: Optimal linear worst-case time complexity
- **Confluence**: proven by CHR confluence checker
- **Correctness**: shown by standard CHR analysis

Other approaches: more code, less logical, less analysis.
Conclusions

Lexicographic Order Constraint Solver in CHR

- Executable specification: short, concise using recursive decomposition and propagation
- Incremental and concurrent: by nature of CHR
- Independent of underlying constraint system
- Complete: propagates as much as possible
- Efficient: Optimal linear worst-case time complexity
- Confluence: proven by CHR confluence checker
- Correctness: shown by standard CHR analysis

Future work:

- Combine with other CHR solvers
- Performance analysis, benchmarking
- Extensions: lex chains, with summation, symmetry breaking
- Automatic derivation from specification possible?