

Towards Logic Programs with Ordered and Unordered Disjunction

Philipp Kärgner¹, Nuno Lopes², Daniel Olmedilla³, Axel Polleres²

¹ L3S Research Center and Leibniz University Hannover

² DERI Galway, Ireland

³ Telefonica R&D, Madrid



Outline

- Ordered and Unordered Disjunction
 - what are they?
 - why do we need both?
- DLPOD (Disjunctive Logic Programs with Ordered Disjunction)
 - potential answer sets \rightarrow Split programs
 - optimal answer sets \rightarrow Pareto optimality wrt. partial orders on literals
- Modeling partial order preferences
- Summary

Ordered and unordered

- “unordered” Disjunction

- extension to classic logic programs:
- `cinema v pub v tv.`
- results in three answer sets `{cinema}`, `{pub}`, `{tv}`



- ordered Disjunction (Brewka et al. 2004):

- LPOD – Logic Programming with Ordered Disjunction
- ASP with special kind of disjunction:
- `cinema x pub x tv.`
- used to express preferences over literals
- yields preferences over answer sets,
in this case `{cinema}` is pref. to `{pub}` is pref. to `{tv}`
- one answer set, namely `{cinema}`,
(since it dominates the others)



Why both, ordered and unordered?

- preference expressions need something in between

no order (DLP)
cinema v pub v tv



total order (LPOD)
cinema x pub x tv



Why both, ordered and unordered?

- preference expressions need something in between

no order (DLP)

cinema \vee pub \vee tv



partial order

cinema \times (pub \vee tv)

cinema is preferred over both, pub and tv, but between pub and tv, there is no preference



total order (LPOD)

cinema \times pub \times tv



DLPOD



DLPOD – Disjunctive Logic Programs with Ordered Disjunction

```
cinema x (pub v tv) ← not sunny.  
beach v park ← sunny.
```

Intuition:

- three 'potential' answer sets: $\{\mathbf{cinema}\}, \{\mathbf{pub}\}, \{\mathbf{tv}\}$
- $\{\mathbf{cinema}\}$ is the most preferred one
- $\{\mathbf{cinema}\}$ is the only actual answer set of the DLPOD

Semantics:

- computing potential answer sets \rightarrow split programs
- which answer set is preferred \rightarrow preference relation on AS of splits

Split programs

```
cinema x (pub v tv) ← not sunny.
```

1. transform head into a normal form (ODNF):

```
(cinema x pub) v (cinema x tv) ← not sunny.
```

2. generate all split programs (containing only \vee):

```
1. cinema v cinema ← not sunny.
```

{cinema}



```
2. cinema v tv ← not sunny, not cinema.
```

{tv}



```
3. pub v cinema ← not sunny, not cinema.
```

{pub}



```
4. pub v tv ← not sunny, not cinema, not cinema.
```

{pub}, {tv}



Comparing Answer Sets

`(cinema x pub) v (cinema x tv) ← not sunny.`

- how to decide which answer set of preferred?
- satisfaction degree vector (SDV) determines how good a particular answer set is wrt. the program
 - `{cinema}` → (1,1)
 - `{pub}` → (2,ε)
 - `{tv}` → (ε,2)
- the answer sets with the Pareto optimal satisfaction degree vectors are answer sets of the DLPOD
- proper generalization of LPOD (SDV is a singleton)

Complexity and Implementation aspects

- Split programs can be arbitrary disjunctive programs
 - existence of an optimal answer set is Σ_2^P -complete
 - whether a candidate AS is optimal is in Π_2^P
 - whether a Literal l in the optimal answer set is in Σ_3^P
- We extended the notion of head-cycle-freeness for DLPODs
 - complexity drops
- Interleaved generator-tester implementation
 - using disj. ASP solvers
 - extending previous algorithms for LPOD [Brewka et al. 2004]
- integrated implementation possible for head-cycle-free DLPODs
 - using results in [Eiter,Polleres,TPLP2006]

An example generator

$$r = (A \times B) \vee (C \times D) \leftarrow \text{Body}.$$

- (a) $1\{c_{r,1}(1), c_{r,1}(2)\}1 \leftarrow \text{Body}.$
 $1\{c_{r,2}(1), c_{r,2}(2)\}1 \leftarrow \text{Body}.$
- (b) $h_{r,1} \vee h_{r,2} \leftarrow b_{r,1}, b_{r,2}, \text{Body}.$
- (c) $h_{r,1} \leftarrow A, c_{r,1}(1). A \leftarrow h_{r,1}, c_{r,1}(1).$
 $h_{r,1} \leftarrow B, c_{r,1}(2). B \leftarrow h_{r,1}, c_{r,1}(2).$
 $h_{r,2} \leftarrow C, c_{r,2}(1). C \leftarrow h_{r,2}, c_{r,2}(1).$
 $h_{r,2} \leftarrow D, c_{r,2}(2). D \leftarrow h_{r,2}, c_{r,2}(2).$
- (d) $b_{r,1} \leftarrow c_{r,1}(1).$
 $b_{r,1} \leftarrow c_{r,1}(2), \text{not } A.$
 $b_{r,2} \leftarrow c_{r,2}(1).$
 $b_{r,2} \leftarrow c_{r,2}(2), \text{not } C.$
- (e) $\leftarrow A, \text{not } c_{r,1}(1), \text{not } B, \text{not } C, \text{not } D.$
 $\leftarrow \text{not } A, B, \text{not } c_{r,1}(2), \text{not } C, \text{not } D.$
 $\leftarrow \text{not } A, \text{not } B, C, \text{not } c_{r,2}(1), \text{not } D.$
 $\leftarrow \text{not } A, \text{not } B, \text{not } C, D, \text{not } c_{r,2}(2).$

- DLPOD is a log. programming language that allows for ordered and unordered disjunction
- a method to compute potential answer sets of such a program
- a preference notion that decides which answer sets are optimal
- But how to model *general* partial orders with a combination of \vee and \times ?

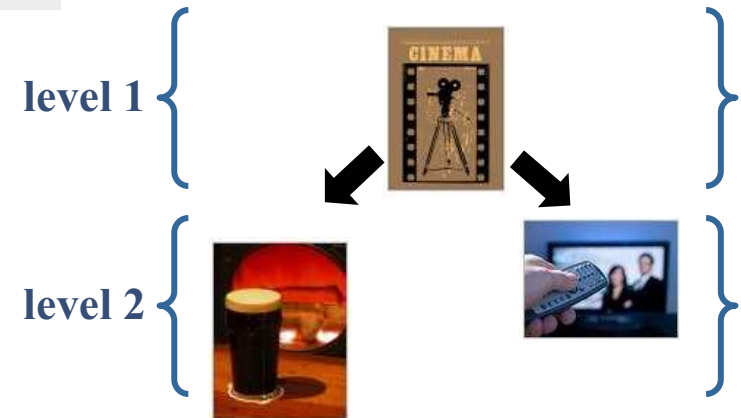


Modeling partial order preferences

- a first try

```
cinema x (pub v tv)
```

put each level of the partial order
in a disjunction and list all levels
with ordered disjunctions:



```
cinema x pub_or_tv.  
pub v tv ← pub_or_tv.
```

Modeling partial order preferences

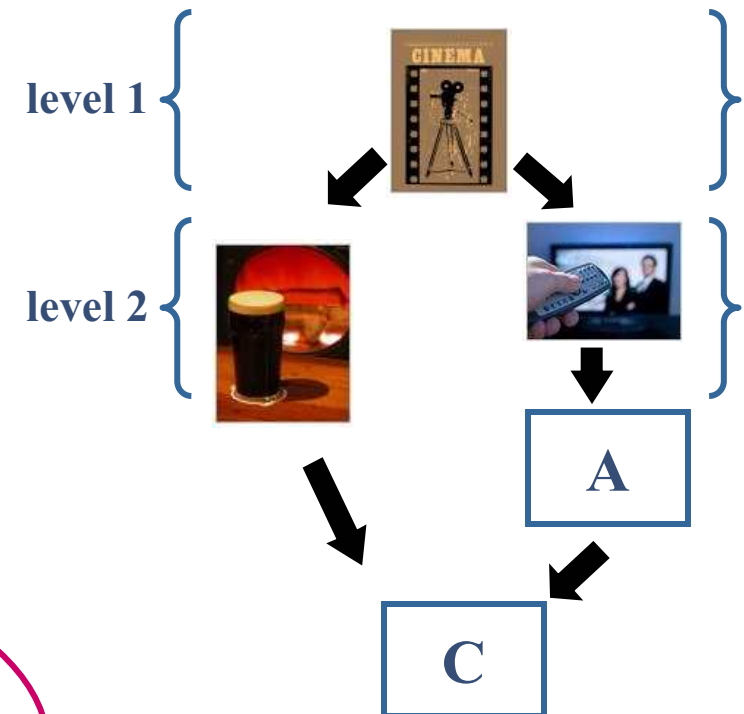
- a first try:

put each level of the partial order
in a disjunction and list all levels
with ordered disjunctions:

`cinema x (pub_or_tv) x A x C`

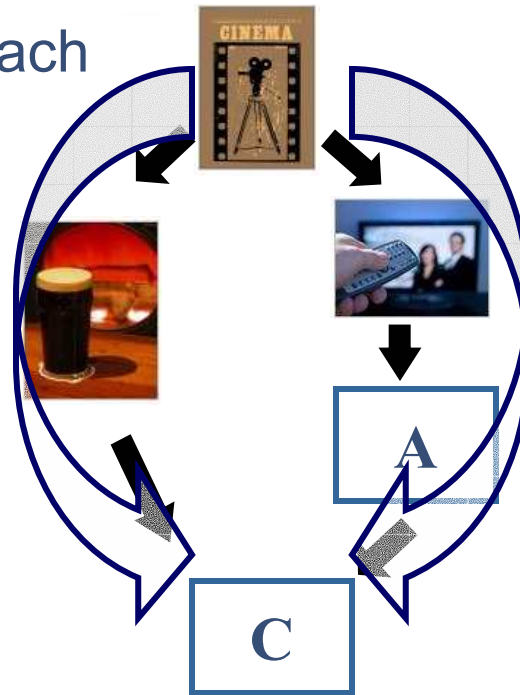
or?

`cinema x tv x (pub_or_A) x C`



Modeling partial orders

- it is needed to encode each path of the partial order:



$(\text{cinema} \times \text{pub} \times \text{C}) \vee (\text{cinema} \times \text{tv} \times \text{A} \times \text{C}) .$



Summary and Outlook

Summary:

- DLPOD allow *qualitative, partial order* preference statements in logic programming
- extension to Brewka et al.'s approach of total ordered disjunction
 - disjunctive split programs
 - fair Pareto preference definition
- DLPODs can be transformed into an interleaved disjunctive logic programs (so normal ASP solvers can handle them)
- Complexity: one level higher than LPOD (for non-head-cycle-free)



Outlook

- semantics without split programs (via reduct)
- proper formalization of distributivity of \times and \vee
- improved transformation to Ordered Disjunctive Normal Form
- remaining hardness proofs
- implementation and evaluation of the algorithms (interleaved and integrated)
- other preference definitions



Thank you for your attention.

Please let me know if there are any questions.

Philipp Kärgen
kaerger@L3S.de

<http://www.L3S.de/~kaerger>