Consistency of Chordal RCC-8 Networks

Michael Sioutis

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

November 8, 2012

Table of Contents

What is Qualitative Spatial Reasoning?

- Qualitative spatial reasoning is based on qualitative abstractions of spatial aspects of the common-sense background knowledge, on which our human perspective on the physical reality is based

Reasons for Qualitative Spatial Reasoning

- Two main reasons why non-precise, qualitative spatial information may be useful:
  1. Only partial information may be available
  2. Spatial constraints are often most naturally stated in qualitative terms

Applications of Qualitative Spatial Reasoning

- Qualitative spatial reasoning is an important subproblem in many *applications*, such as:
  - Robotic navigation
  - High level vision
  - Geographical information systems (GIS)
  - Reasoning and querying with semantic geospatial query languages (e.g., stSPARQL, GeoSPARQL)

Region Connection Calculus

- The Region Connection Calculus (RCC) is a first-order language for representation of and reasoning about topological relationships between extended spatial regions
- RCC abstractly describes regions, that are non-empty regural subsets of some topological space which do not have to be internally connected

The RCC-8 Calculus

- RCC-8 is a *constraint* language formed by the combination of the following eight jointly exhaustive and pairwise disjoint *base relations*:
  - disconnected (DC)
  - externally connected (EC)
  - equal (EQ)
  - partially overlapping (PO)
  - tangential proper part (TPP)
  - tangential proper part inverse (TPPi)
  - non-tangential proper part (NTPP)
  - non-tangential proper part inverse (NTPPi)

The Eight Basic Relations of the RCC-8 Calculus

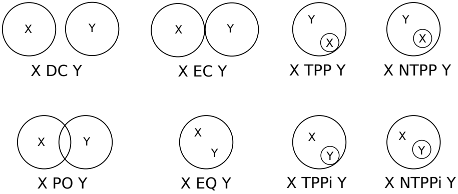


Figure: Two dimesional examples for the eight base relations of RCC-8

From these basic relations, combinations can be built. For example, proper part (PP) is the union of TPP and NTPP.

The RCC-8 Composition Table

◇	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPPi**	**NTPPi**	**EQ**
DC	*	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC	DC	DC
EC	DC,EC PO,TPPi NTPPi	DC,EC PO,TPP TPPi,EQ	DC,EC PO,TPP NTPP	EC,PO TPP NTPP	PO TPP NTPP	DC,EC	DC	EC
PO	DC,EC PO,TPPi NTPPi	DC,EC PO,TPPi NTPPi	*	PO TPP NTPP	PO TPP NTPP	DC,EC PO,TPPi NTPPi	DC,EC PO,TPPi NTPPi	PO
TPP	DC	DC,EC	DC,EC PO,TPP NTPP	TPP NTPP	NTPP	DC,EC PO,TPP TPPi,EQ	DC,EC PO,TPPi NTPPi	TPP
NTPP	DC	DC	DC,EC PO,TPP NTPP	NTPP	NTPP	DC,EC PO,TPP NTPP	*	NTPP
TPPi	DC,EC PO,TPPi NTPPi	EC,PO TPPi NTPPi	PO TPPi NTPPi	PO,EQ TPP TPPi	PO TPP NTPP	TPPi NTPPi	NTPPi	TPPi
NTPPi	DC,EC PO TPPi NTPPi	PO TPPi NTPPi	PO TPPi NTPPi	PO TPPi NTPPi	PO,TPP NTPP NTPPi TPPi,EQ	NTPPi	NTPPi	NTPPi
EQ	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ

The RSAT Reasoning Problem

- RSAT in the RCC-8 framework is the reasoning problem of deciding whether there is a spatial configuration where the relations between the regions can be described by a spatial formula Θ
- RSAT is NP-Complete!
- However, tractable subsets S of RCC-8 exist, such as $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, $\mathcal{Q}_8$ [5], for which the consistency problem can be decided in polynomial time

The RSAT Reasoning Problem

- RSAT in the RCC-8 framework is the reasoning problem of deciding whether there is a spatial configuration where the relations between the regions can be described by a spatial formula Θ
- RSAT is NP-Complete!
- However, tractable subsets S of RCC-8 exist, such as $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, $\mathcal{Q}_8$ [5], for which the consistency problem can be decided in polynomial time

The RSAT Reasoning Problem

- RSAT in the RCC-8 framework is the reasoning problem of deciding whether there is a spatial configuration where the relations between the regions can be described by a spatial formula Θ
- RSAT is NP-Complete!
- However, tractable subsets S of RCC-8 exist, such as $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, $\mathcal{Q}_8$ [5], for which the consistency problem can be decided in polynomial time

Path Concistency

- Approximates consistency and realizes *forward checking* in a backtracking algorithm
- Checks the consistency of triples of relations and eliminates relations that are impossible though iteravely performing the operation

$$R_{ij} \leftarrow R_{ij} \cap R_{ik} \diamond R_{kj}$$

  until a fixed point $\overline{R}$ is reached
- If $R_{ij} = \emptyset$ for a pair (i, j) then R is inconsistent, otherwise $\overline{R}$ is *path-consistent*.
- Computing $\overline{R}$ is done in $O(n^3)$

About PyRCC8..

- PyRCC8[1] is an efficient qualitative spatial reasoner written in pure Python. It employs PyPy[2], a fast, compliant implementation of the Python 2 language

- PyRCC8 offers a path consistency algorithm for solving tractable RCC-8 networks and a backtracking-based algorithm for general networks

[1] http://pypi.python.org/pypi/PyRCC8

[2] http://pypy.org/

Comparing PC Implementations of Different Reasoners

- We compare the PC implementation of PyRCC8 to the PC implementations of the following qualitative spatial reasoners:
  - Renz's solver[3]
  - GQR[4]
  - Pellet Spatial[5]
- The admingeo[6] dataset (11761 regions / 77910 relations) was used which was properly translated to fit the input format of the different PC implementations

---

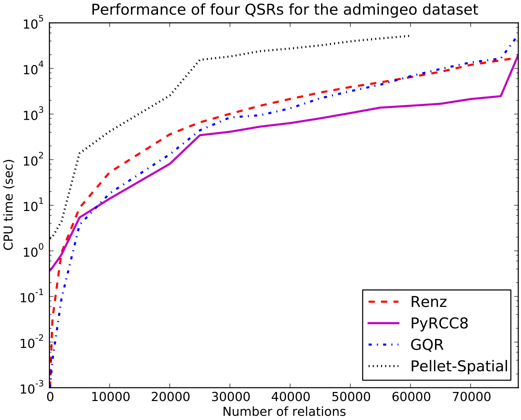
[3] http://users.rsise.anu.edu.au/%7Ejrenz/software/rcc8-csp-solving.tar.gz

[4] http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Tools/gqr.html

[5] http://clarkparsia.com/pellet/spatial/

[6] http://data.ordnancesurvey.co.uk/ontology/admingeo/

Evaluation with a Large Dataset



Consistency

- To explore the search space for the general case of RCC-8 networks in order to solve an instance Θ of RSAT, some sort of backtracking must be used
- We implemented two backtracking algorithms:
  1. A strictly recursive one
  2. An equivalent iterative one which resembles recursion

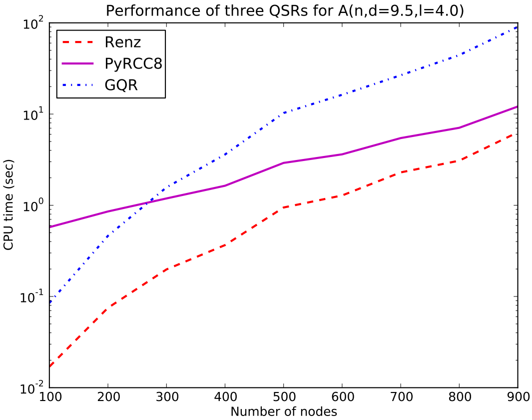
Comparing PyRCC8 to Other Reasoners

- We compare PyRCC8 to the following qualitative spatial reasoners:
  - Renz's solver[7]
  - GQR[8]
- Different size n of instances from $A(n, d = 9.5, l = 4.0)$ were used

[7] http://users.rsise.anu.edu.au/%7Ejrenz/software/rcc8-csp-solving.tar.gz

[8] http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Tools/gqr.html

Comparison Diagram



$\bigtriangledown$-Path Concistency

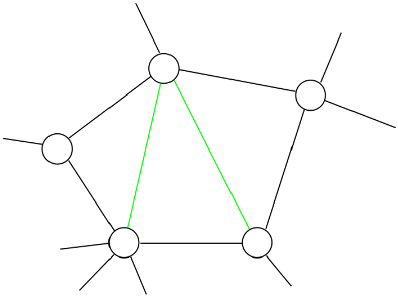
- Up till now, all aproaches in qualitative spatial reasoning enforce path consistency on a complete spatial network
- We propose enforcing path consistency on a *chordal* spatial network [2] as Chmeiss and Condotta have done for temporal networks [3], and we call this type of local consistency as $\bigtriangledown$*-path consistency* for clarity

$\bigtriangledown$-Path Concistency

- Up till now, all aproaches in qualitative spatial reasoning enforce path consistency on a complete spatial network
- We propose enforcing path consistency on a *chordal* spatial network [2] as Chmeiss and Condotta have done for temporal networks [3], and we call this type of local consistency as *$\bigtriangledown$-path consistency* for clarity

Chordal Graph

- A graph is chordal if each of its cycles of four or more nodes has a *chord*, which is an edge joining two nodes that are not adjacent in the cycle
- An example of a chordal graph is shown below:



Triangulation

- Triangulation of a given graph is done by eliminating the vertices one by one and connecting all vertices in the neighbourhood of each eliminated vertex with *fill edges*
- Determining a minimum triangulation is an NP-hard problem
- Use of several heuristics for sub-optimal solutions (e.g. minimum degree, minimum fill)
- Chordality checking can be done efficiently in $O(|V| + |E|)$ time, for a graph $G = (V, E)$ (e.g., with MCS, LexBFS)

Preliminaries

- Let $G = (V, E)$ be an undirected chordal graph. There exists a tree T, called a *clique tree* of G, whose vertex set is the set of maximal cliques of G
- Let C be a constraint network from a given CSP. Then, $\mathcal{V}_C$ refers to the set of variables of C
- If $\mathcal{V}$ is any set of variables, $C_{\mathcal{V}}$ will be the constraint network C that involves variables of $\mathcal{V}$

Patchwork Property in RCC-8 Networks

Definition

We will say that a CSP has the patchwork property if for any finite satisfiable constraint networks C *and* C' *of the CSP such that* $C_{\mathcal{V}_C \cap \mathcal{V}_{C'}} = C'_{\mathcal{V}_C \cap \mathcal{V}_{C'}}$*, the constraint network* $C \cup C'$ *is satisfiable [4].*

Proposition

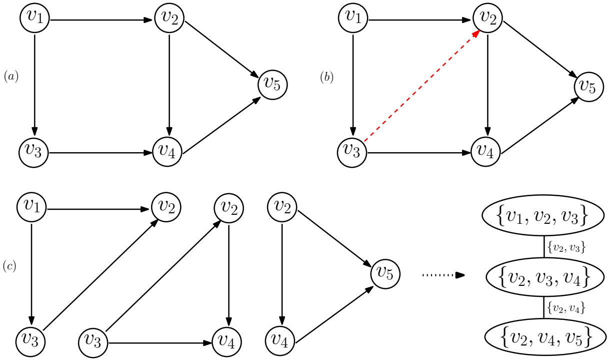
The three CSPs for path consistent $\hat{\mathcal{H}}_8, \mathcal{C}_8,$ *and* $\mathcal{Q}_8$ *networks, respectively, all have patchwork [4].*

Proposition

Proposition

Let C be an RCC-8 constraint network with relations from $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ on its edges. Let G be the chordal graph that results from triangulating the associated constraint graph of C, and T a clique tree of G. C is consistent if all the networks corresponding to the nodes of T are path consistent.

Example


(a)
v1
v2
v3
v4
v5
(b)
v1
v2
v3
v4
v5
(c)
v1
v2
v3
v2
v3
v4
v2
v4
v5
{v1, v2, v3}
{v2, v3}
{v2, v3, v4}
{v2, v4}
{v2, v4, v5}

PyRCC8$\triangledown$

- PyRCC8$\triangledown$ is a chordal reasoner which was developed by extending PyRCC8
- Similarly to PyRCC8, PyRCC8$\triangledown$ offers a $\triangledown$-path consistency algorithm for solving tractable RCC-8 networks and a backtracking-based algorithm for general networks

$\triangledown$-Path Consistency Algorithm

$\triangledown$-**Path-Consistency**(C, G)
Input: A constraint network C and its chordal graph G
Output: True or False

Q $\leftarrow$ {(i, j) | (i, j) $\in$ E } // Initialize the queue
while Q is not empty **do**
    select and delete an (i, j) from Q
    for each k such that (i, k), (k, j) $\in$ E **do**
        t $\leftarrow$ $C_{ik} \cap (C_{ij} \diamond C_{jk})$
        if t $\neq$ C_{ik} **then**
            if t = $\emptyset$ **then**
                return False
            $C_{ik} \leftarrow$ t
            $C_{ki} \leftarrow \breve{t}$
            Q $\leftarrow$ Q $\cup$ {(i, k)}
        t $\leftarrow$ $C_{kj} \cap (C_{ki} \diamond C_{ij})$
        if t $\neq$ C_{kj} **then**
            if t = $\emptyset$ **then**
                return False
            $C_{kj} \leftarrow$ t
            $C_{jk} \leftarrow \breve{t}$
            Q $\leftarrow$ Q $\cup$ {(k, j)}
return True

Complexity Analysis

- Let δ denote the maximum degree of a vertex of G
- For each arc (i, j) selected at line 3, we have at most δ vertices of G corresponding to index k such that v_i, v_j, v_k forms a triangle
- Additionaly, there exist $|E|$ arcs in the network and one can remove at most $|\mathcal{B}|$[9] values from any relation that corresponds to an arc
- It results that the time complexity of $\bigtriangledown$-path consistency is $O(\delta \cdot |E| \cdot |\mathcal{B}|)$

[9] $\mathcal{B}$ refers to the set of base relations of RCC-8

Recursive ▽-Consistency Algorithm

▽-**Consistency**(C, G)
Input: A constraint network C and its chordal graph G
Output: A refined constraint network C' if C is satisfiable or None

if not ▽-**Path-Consistency**(C, G) **then**
  return **None**
if no constraint can be split **then**
  return **C**
else
  choose unprocessed constraint $x_i R x_j$; split R into $S_1, \ldots, S_k \in S$: $S_1 \cup \ldots \cup S_k = R$
  Values $\leftarrow \{S_l \mid 1 \leq l \leq k\}$
for V in Values **do**
  replace $x_i R x_j$ with $x_i V x_j$ in C
  result = ▽-**Consistency**(C, G)
  if result $\neq$ **None** **then**
    return result
return **None**

Iterative ▽-Consistency Algorithm

▽-**Consistency**(C, G)
Input: A constraint network C, A chordal graph G
Output: A refined constraint network C' if C is satisfiable or None

```
Stack ← {} // Initialize stack
if not ▽-Path-Consistency(C, G) then
    return None
while 1 do
    if no constraint can be split then
        return C
    else
        choose unprocessed constraint x_i R x_j ; split R into S_1, ..., S_k ∈ S: S_1 ∪ ... ∪ S_k = R
        Values ← {S_l | 1 ≤ l ≤ k}
    while 1 do
        if not Values then
            while Stack do
                C, Values = Stack.pop()
                if Values then
                    break
            else
                return None
        V = Values.pop()
        replace x_i R x_j with x_i V x_j in C
        if ▽-Path-Consistency(C, G) then
            break
    Stack.push(C, Values)
raise RuntimeError, Can't happen
```

Comparing PyRCC8$\triangledown$ to PyRCC8

- We compare PyRCC8$\triangledown$ to PyRCC8, a complete graph dedicated reasoner, using the following data:
  - *Random instances* composed from the set of all RCC-8 relations
  - The *admingeo*[10] *dataset*

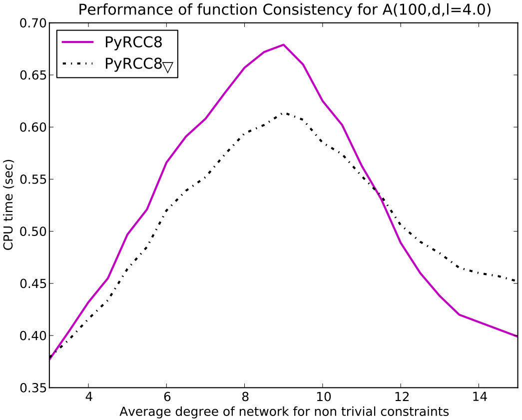
[10] `http://data.ordnancesurvey.co.uk/ontology/admingeo/`

Experimenting with Random Instances

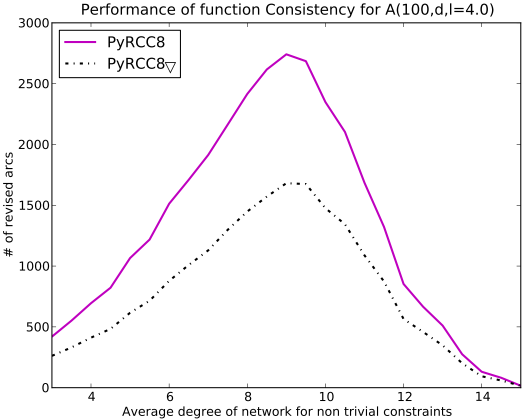
- We generated instances from $A(100, d, l = 4.0)$, for d varying from 3 to 15 with a step of 0.5. For each series, 300 networks were generated using Renz's network generator[11]
- We used the Horn relations set as our split set, and the dynamic/local constraint scheme with a weighted queue configuration, since it proved to be the best combination for both reasoners, confirming the results in [5]

[11] http://users.rsise.anu.edu.au/%7Ejrenz/software/rcc8-csp-solving.tar.gz

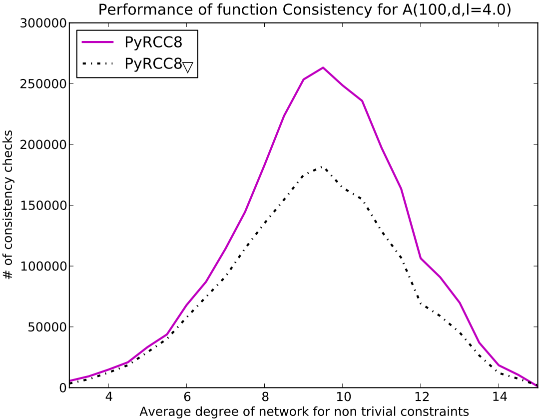
Comparison Diagram on CPU time



Comparison Diagram on # of Revised Arcs



Comparison Diagram on # of Checked Constraints



Results Summary

	PyRCC8	PyRCC8▽	%
CPU time	0.524s	0.509s	2.80%
revised arcs	1300.681	801.204	38.40%
checked constraints	105751.173	74864.985	29.21%

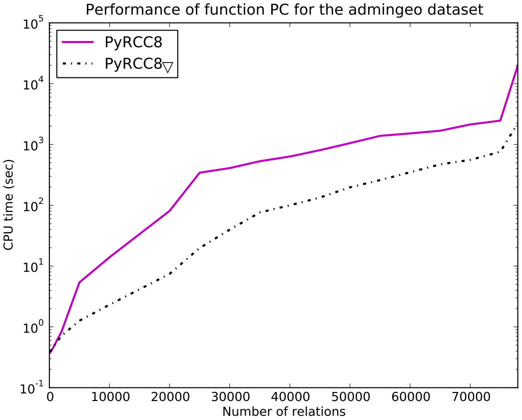
Table: Comparison based on the average of different parameters

Experimenting with the Admingeo Dataset

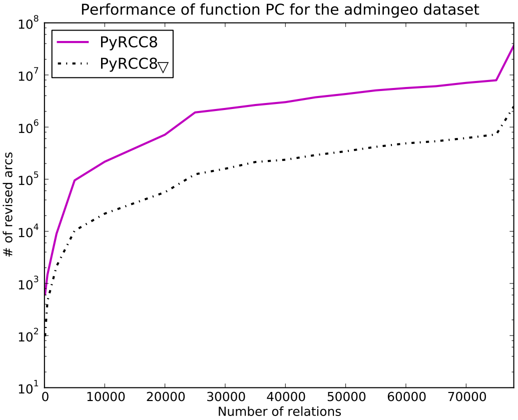
- The *admingeo*[12] *dataset* consists of 11761 regions and 77910 base relations, thus being an extremely large and sparse network, making itself a good candidate for stress testing different path consistency implementations
- We used a simple queue configuration, since the weighted variants made no difference on this dataset other than using much more memory

12 http://data.ordnancesurvey.co.uk/ontology/admingeo/

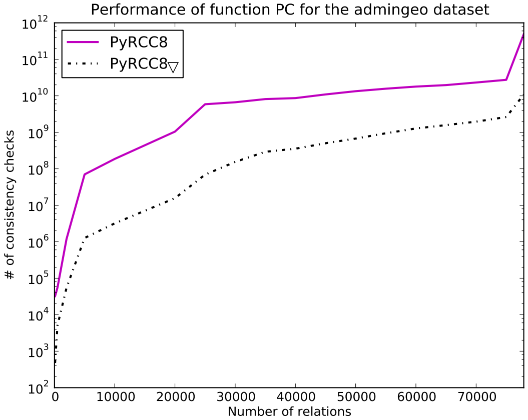
Comparison Diagram on CPU Time



Comparison Diagram on # of Revised Arcs



Comparison Diagram on # of Checked Constraints



Results Summary

	PyRCC8	PyRCC8▽	%
CPU time	1825.129s	289.203s	84.15%
revised arcs	4834133.78	373080.28	92.28%
checked constraints	$3.606e+10$	$1.181e+09$	96.72%

Table: Comparison based on the average of different parameters

Test Machine

- All experiments were carried out on a computer with an Intel Xeon 4 Core X3220 processor with a CPU frequency of 2.40 GHz, 8 GB RAM, and the Debian Lenny x86 64 OS
- Renz's solver and GQR were compiled with gcc/g++ 4.4.3
- PelletSpatial was run with OpenJDK 6 build 19, which implements Java SE 6
- PyRCC8 was run with PyPy 1.8, which implements Python 2.7.2
- Only one of the CPU cores was used for the experiments

Conclusions

- We made the case for a new generation of RCC-8 reasoners implemented in Python, and making use of advanced Python environments, such as PyPy, utilizing trace-based JIT compilation techniques
- We introduced $\triangledown$-path consistency for RCC-8 networks
- We showed that $\triangledown$-path consistency is sufficient to decide the consistency problem for the maximal tractable subsets $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ of RCC-8
- We implemented a chordal graph dedicated reasoner for RCC-8 networks
- We showed expirimentally that $\triangledown$-path consistency can offer a great advantage over full path consistency on sparse graphs

Main Points

- Explore self learning heuristics regarding variable and value selection
- Create module to generate spatial CSPs
- Transform PyRCC8 into a generic qualitative reasoner
- Use other methods of triangulation and compare the behavior of partial path consistency for these different methods
- Perform experiments with other possible real datasets, such as GADM[13]

[13] http://gadm.geovocab.org/

Acknowledge

- This work was funded by the FP7 project TELEIOS (257662)
- I would also like to thank my colleagues, and Katia Papakonstantinopoulou especially, for their help, interest, and advice

References

[Van Beek and Manchak]
The design and experimental analysis of algorithms for temporal reasoning
JAIR, vol. 4, pages 1–18, 1996

[Bliek and Sam-Haroud]
Path Consistency on Triangulated Constraint Graphs
In *IJCAI*, 1999

[Chmeiss and Condotta]
Consistency of Triangulated Temporal Qualitative Constraint Networks
In *ICTAI*, 2011

[Huang]
Compactness and Its Implications for Qualitative Spatial and Temporal Reasoning
In *KR*, 2012

[Renz and Nebel]
Efficient Methods for Qualitative Spatial Reasoning
JAIR, vol. 15, pages 289–318, 2001

The End

Any Questions?