Principles of Knowledge Representation and Reasoning

3. Qualitative Representation and Reasoning

3.8 Double Cross: A Calculus for Qualitative Navigation

Bernhard Nebel

- Reminder: Computationally well-behaved calculi
- Relating three points in the plane
- Generalizing relation algebras
- Closing the Double Cross under Permutation and Composition
- Computational Complexity
- Outlook & Open Problems
Computationally Well-Behaved Calculi

The calculi we looked at so far appeared to be well-behaved
Computationally Well-Behaved Calculi

The calculi we looked at so far appeared to be well-behaved

**Closedness**: If the set of relations is **not closed** under intersection, composition, and converse, there are two possibilities
Computationally Well-Behaved Calculi

The calculi we looked at so far appeared to be well-behaved.

**Closedness**: If the set of relations is not closed under intersection, composition, and converse, there are two possibilities:

- we have to make more distinctions and finally arrive at a finite set that is closed under intersection, composition, and converse.
The calculi we looked at so far appeared to be well-behaved.

**Closedness**: If the set of relations is not closed under intersection, composition, and converse, there are two possibilities:

- we have to make more distinctions and finally arrive at a finite set that is closed under intersection, composition, and converse.
- there exists no finite set of atomic relations.
Computationally Well-Behaved Calculi

The calculi we looked at so far appeared to be well-behaved.

**Closedness:** If the set of relations is not closed under intersection, composition, and converse, there are two possibilities:

- we have to make more distinctions and finally arrive at a finite set that is closed under intersection, composition, and converse.
- there exists no finite set of atomic relations.

**Path-consistency method decides large fragments:** Is something we usually have (ORD-Horn, $\mathcal{H}_8$). At least the fragment containing all base relations and the universal relation is usually decided by the PC method.
Computationally Well-Behaved Calculi

The calculi we looked at so far appeared to be well-behaved

**Closedness:** If the set of relations is not closed under intersection, composition, and converse, there are two possibilities:

- we have to make more distinctions and finally arrive at a finite set that is closed under intersection, composition, and converse.
- there exists no finite set of atomic relations

**Path-consistency method decides large fragments:** Is something we usually have (ORD-Horn, $\mathcal{H}_8$). At least the fragment containing all base relations and the universal relation is usually decided by the PC method.

**Path-consistency method decides CSP with only base relations:** If we do not have even this, the satisfiability problem might not be in NP!
A Motivating Example

1. From \( a \) go to \( b \) and make a right turn aiming forward to a point \( c \).
A Motivating Example

1. From $a$ go to $b$ and make a **right turn aiming forward** to a point $c$.

2. From $b$ goto to $c$ and make a **(perpendicular) right turn** going to $d$. 

![Diagram showing the path from a to b, then right turn aiming forward to c, then (perpendicular) right turn going to d.](image-url)
A Motivating Example

1. From $a$ go to $b$ and make a **right turn aiming forward** to a point $c$.

2. From $b$ goto to $c$ and make a (perpendicular) **right turn** going to $d$.

From these two descriptions, it is possible to infer that $a$ and $d$ **cannot be identical**.
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).

![Diagram showing the Double-Cross relationship between points a, b, and c.](attachment:diagram.png)
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).

![Diagram of Double-Cross]

- $l_f$  
- $s_f$  
- $r_f$  
- $l_p$  
- $s_p$  
- $r_p$  
- $b$  
- $s_l$  
- $a$
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).

\[
\begin{array}{ccc}
  & sf & rf \\
lp & & \\
  sp & & rp \\
  & lc & sc & rc \\
  & & a \\
\end{array}
\]
**The Double-Cross Calculus**

**Double-Cross**: relating three points (describing a path in a qualitative way).
Double-Cross: relating three points (describing a path in a qualitative way).

\[ \begin{align*}
& l_f & & s_f & & r_f \\
\hline
& l_p & & s_p & & r_p \\
& l_c & & s_c & & r_c \\
\hline
& l_l & & s_l & & r_l \\
& l_b & & s_b & & r_b
\end{align*} \]
The Double-Cross Calculus

**Double-Cross**: relating three points (describing a path in a qualitative way).

*Note*: These relations are not exhaustive.
**The Double-Cross Calculus**

**Double-Cross**: relating three points (describing a path in a qualitative way).

Note: These relations are not exhaustive

\( \sim \) Add two relations: \( eq \) (all three points are identical) and \( ex \ (a = b) \)
Double-Cross Constraint Systems

- We have ternary relations between points in $\mathbb{R}^2$. 
Double-Cross Constraint Systems

• We have ternary relations between points in $\mathbb{R}^2$.

• Since the relations are ternary, we have to generalize the path-consistency method to ternary relations.
Double-Cross Constraint Systems

- We have **ternary** relations between points in $\mathbb{R}^2$.

- Since the relations are **ternary**, we have to generalize the **path-consistency method** to ternary relations.

- Converse $\sim$ **Permutations**
Double-Cross Constraint Systems

- We have ternary relations between points in $\mathbb{R}^2$.

- Since the relations are ternary, we have to generalize the path-consistency method to ternary relations.

- Converse $\sim$ Permutations ($3! = 6$)
Double-Cross Constraint Systems

- We have ternary relations between points in $\mathbb{R}^2$.

- Since the relations are ternary, we have to generalize the path-consistency method to ternary relations.

- Converse $\leadsto$ Permutations ($3! = 6$)

- Composition $\leadsto$ compositions over different arguments
Double-Cross Constraint Systems

- We have *ternary* relations between points in $\mathbb{R}^2$.

- Since the relations are *ternary*, we have to generalize the path-consistency method to ternary relations.

- Converse $\leadsto$ **Permutations** ($3! = 6$)

- Composition $\leadsto$ **compositions over different arguments**

- Closure under these new operations gives hopefully some form of local consistency.
Permutation Operations

The $3!$ ways of exchanging arguments
Permutation Operations

The $3!$ ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>$a, b, c$</td>
</tr>
</tbody>
</table>
The $3!$ ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>$a, b, c$</td>
</tr>
<tr>
<td>inversion</td>
<td>INV</td>
<td>$b, a, c$</td>
</tr>
</tbody>
</table>
Permutation Operations

The $3!$ ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>$a, b, c$</td>
</tr>
<tr>
<td>inversion</td>
<td>INV</td>
<td>$b, a, c$</td>
</tr>
<tr>
<td>short cut</td>
<td>Sc</td>
<td>$a, c, b$</td>
</tr>
</tbody>
</table>
### Permutation Operations

The 3! ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>a, b, c</td>
</tr>
<tr>
<td>inversion</td>
<td>INV</td>
<td>b, a, c</td>
</tr>
<tr>
<td>short cut</td>
<td>Sc</td>
<td>a, c, b</td>
</tr>
<tr>
<td>inverse short cut</td>
<td>SCI</td>
<td>c, a, b</td>
</tr>
</tbody>
</table>
### Permutation Operations

The $3!$ ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>$a, b, c$</td>
</tr>
<tr>
<td>inversion</td>
<td>INV</td>
<td>$b, a, c$</td>
</tr>
<tr>
<td>short cut</td>
<td>Sc</td>
<td>$a, c, b$</td>
</tr>
<tr>
<td>inverse short cut</td>
<td>SCI</td>
<td>$c, a, b$</td>
</tr>
<tr>
<td>homing</td>
<td>HM</td>
<td>$b, c, a$</td>
</tr>
</tbody>
</table>
## Permutation Operations

The $3!$ ways of exchanging arguments:

<table>
<thead>
<tr>
<th>term</th>
<th>symbol</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>identical</td>
<td>ID</td>
<td>$a, b, c$</td>
</tr>
<tr>
<td>inversion</td>
<td>INV</td>
<td>$b, a, c$</td>
</tr>
<tr>
<td>short cut</td>
<td>Sc</td>
<td>$a, c, b$</td>
</tr>
<tr>
<td>inverse short cut</td>
<td>SCI</td>
<td>$c, a, b$</td>
</tr>
<tr>
<td>homing</td>
<td>HM</td>
<td>$b, c, a$</td>
</tr>
<tr>
<td>inverse homing</td>
<td>HMI</td>
<td>$c, b, a$</td>
</tr>
</tbody>
</table>
We have $Sc(If) \subseteq rc.$
Short Cut Leads to New Relations

We have $\text{Sc}(lf) \subseteq rc$.  

However, we have $rc \not\subseteq \text{Sc}(lf)$.
Short Cut Leads to New Relations

We have $\text{Sc}(lf) \subseteq rc$.

However, we have $rc \not\subseteq \text{Sc}(lf)$:
The Circle on the Double Cross
The Circle on the Double Cross
The Circle on the Double Cross
The Circle on the Double Cross
The Circle on the Double Cross

New relation: $orc$ (on right circle)
The Circle on the Double Cross

New relation: \(orc (on \ right \ circle)\)

\[orc \subseteq Sc(lp)\]
Add New Relations in Order to Close the Relation System

- Distinguish between “close” and far “points”
Add New Relations in Order to Close the Relation System

- Distinguish between “close” and far “points”:

<table>
<thead>
<tr>
<th>If</th>
<th>sf</th>
<th>rf</th>
</tr>
</thead>
<tbody>
<tr>
<td>lp</td>
<td>sp</td>
<td>rp</td>
</tr>
</tbody>
</table>

```
elc
ilc
irc
erc
ilc
erc
eq + ex
```
Add New Relations in Order to Close the Relation System

- Distinguish between “close” and far “points”:

```
lf sf rf
lp sp rp
ll sl rl
lb sb rb
elc orcolc ilc irc erc
```

- This refined set of 21 base relations is **closed under permutations**!
Compositions (1)

- We have to consider also compositions of relations
Compositions (1)

- We have to consider also *compositions* of relations

- Example:

```
- - - a  - - - -
  ^            ^
  |            |  c
  |            |
  b ----> d
```

```
- - - - a  - - -
  |      |    ^
  |      |    |
  - - - - c  - -
```

```
- - - - b  - - -
  |      |    ^
  |      |    |
  - - - - c  - -
```

```
- - - - - a  - - -
  |      |    ^
  |      |    |
  - - - - - c  - -
```
Compositions (1)

- We have to consider also compositions of relations
- Example:
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)
In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)

In ternary relation systems, there are many ways to compose two relations
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)
- In ternary relation systems, there are many ways to compose two relations
  ~⇒ one argument sharing compositions
Compositions (2)

• In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations).

• In ternary relation systems, there are many ways to compose two relations:
  - one argument sharing compositions
  - no restrictions on the arguments!
  - two arguments sharing compositions, e.g.

\[ r_1(a, b, c), r_2(a, c, d) \sim r'(a, b, d) \]
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations).

- In ternary relation systems, there are many ways to compose two relations:
  - one argument sharing compositions
  - no restrictions on the arguments!
  - two arguments sharing compositions, e.g.

\[ r_1(a, b, c), r_2(a, c, d) \leadsto r'(a, b, d) \quad (\exists c)(\forall a, b, d) \]
Compositions (2)

• In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)

• In ternary relation systems, there are many ways to compose two relations

  $\sim$ one argument sharing compositions $\sim$ no restrictions on the arguments!

  $\sim$ two arguments sharing compositions, e.g.

  $$r_1(a, b, c), r_2(a, c, d) \sim r'(a, b, d) (\exists c)(\forall a, b, d)$$

→ How many ways are there?
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)

- In ternary relation systems, there are many ways to compose two relations

  - one argument sharing compositions → no restrictions on the arguments!

  - two arguments sharing compositions, e.g.

    \[ r_1(a, b, c), r_2(a, c, d) \sim r'(a, b, d) \ (\exists c)(\forall a, b, d) \]

  → How many ways are there?
  → 6 ways to pick two arguments out of three with changing order
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations)

- In ternary relation systems, there are many ways to compose two relations
  - one argument sharing compositions
  - no restrictions on the arguments!
  - two arguments sharing compositions, e.g.
    
    \[ r_1(a, b, c), r_2(a, c, d) \leadsto r'(a, b, d) \ (\exists c)(\forall a, b, d) \]

→ How many ways are there?

→ 6 ways to pick two arguments out of three with changing order, same for second relation
Compositions (2)

- In binary relations systems, there is just one sensible way to compose relations (four, if we allow for conversions in the original relations).

- In ternary relation systems, there are many ways to compose two relations:
  - one argument sharing compositions
  - no restrictions on the arguments!
  - two arguments sharing compositions, e.g.

\[
r_1(a, b, c), r_2(a, c, d) \leadsto r'(a, b, d) \quad (\exists c)(\forall a, b, d)
\]

→ How many ways are there?

→ 6 ways to pick two arguments out of three with changing order, same for second relation, i.e., 36 different compositions.
Compositions (3)

- **One type of composition** is enough to generate all other compositions (using permutations)
One type of composition is enough to generate all other compositions (using permutations)

∀a, b, d: (r₁ ◦ r₂)(a, b, d) ↔ ∃c: r₁(a, b, c) ∧ r₂(a, c, d)
Compositions (3)

- **One type of composition** is enough to generate all other compositions (using permutations)

\[ \forall a, b, d: (r_1 \bowtie r_2)(a, b, d) \leftrightarrow \exists c: r_1(a, b, c) \land r_2(a, c, d) \]

- Using this one, we can generate the rest
Compositions (3)

- **One type of composition** is enough to generate all other compositions (using permutations)
  \[ (r_1 \circ r_2)(a, b, d) \iff \exists c: r_1(a, b, c) \land r_2(a, c, d) \]

- Using this one, we can generate the rest

- For instance, \[ \forall a, b, d: r_1(c, a, b) \land r_2(a, d, c) \iff r'(a, b, d) \]
Compositions (3)

- **One type of composition** is enough to generate all other compositions (using permutations)

\[ \forall a, b, d: (r_1 \odot r_2)(a, b, d) \leftrightarrow \exists c: r_1(a, b, c) \land r_2(a, c, d) \]

- Using this one, we can generate the rest

- For instance, \( \forall a, b, d: r_1(c, a, b) \land r_2(a, d, c) \leftrightarrow r'(a, b, d) \)

\[ r' = \text{HM}(r_1) \odot \text{Sc}(r_2) \]
Closure under Composition?

Consider the composition $olc \diamond olc$. 
Closure under Composition?

Consider the composition $olc \diamond olc$. 
Consider the composition $\text{olc} \diamond \text{olc}$.

$$(\text{olc} \diamond \text{olc}) \subseteq \{ll, lb, lc\}, \text{ however } (\text{olc} \diamond \text{olc}) \not\supseteq \{ll, lb, lc\}.$$
Closure under Composition?

Consider the composition $olc \odot olc$.

$$(olc \odot olc) \subseteq \{ll, lb, lc\}, \text{ however } (olc \odot olc) \not\subseteq \{ll, lb, lc\}.$$ 

Can we complete the relation system?
Consider the relation: \((\text{olc} \lozenge \text{olc}) \cap \text{ll}\)
Consider the relation: \((olc \diamond olc) \cap II\)
Refining the II relation

Consider the relation: \((\text{olc} \Diamond \text{olc}) \cap \text{II}\)

Denote this relation by \(\text{lcose}[0.5]\).
Refining the \(\|\) relation

Consider the relation: \((\text{olc} \bowtie \text{olc}) \cap \|\)

Denote this relation by \(\text{lc}\text{close}[0.5]\).

\(\text{rc}\text{close}[0.5]\) is defined in a similar way.
Infinite Relation Systems

**Theorem.** There exists no finite refinement of the double-cross relation system that is closed under permutation, composition, and intersection.
Infinite Relation Systems

**Theorem.** There exists no finite refinement of the double-cross relation system that is closed under permutation, composition, and intersection.

**Proof Sketch.** We show that if there exists the relation $lclose[r/2]$, for $0 < r \leq 1$, there exists also the relation $lclose[r/8]$. 
Theorem. There exists no finite refinement of the double-cross relation system that is closed under permutation, composition, and intersection.

Proof Sketch. We show that if there exists the relation $lclose[r/2]$, for $0 < r \leq 1$, there exists also the relation $lclose[r/8]$. This means we can construct the relation sequence $lclose[1/2], lclose[1/8], lclose[1/32], \ldots$
**Theorem.** There exists no finite refinement of the double-cross relation system that is closed under permutation, composition, and intersection.

**Proof Sketch.** We show that if there exists the relation $lclose[r/2]$, for $0 < r \leq 1$, there exists also the relation $lclose[r/8]$. This means we can construct the relation sequence $lclose[1/2], lclose[1/8], lclose[1/32], \ldots$

Consider the relation $((lclose[r/2] \diamond rclose[0.5]) \cap sc)(a, b, d)$
Infinite Relation Systems

**Theorem.** There exists no finite refinement of the double-cross relation system that is closed under permutation, composition, and intersection.

**Proof Sketch.** We show that if there exists the relation $\text{lclose}[r/2]$, for $0 < r \leq 1$, there exists also the relation $\text{lclose}[r/8]$. This means we can construct the relation sequence $\text{lclose}[1/2], \text{lclose}[1/8], \text{lclose}[1/32], \ldots$

Consider the relation $((\text{lclose}[r/2] \diamond \text{rclose}[0.5]) \cap \text{sc})(a, b, d)$:

![Diagram showing points a, b, c, d connected by circles and lines illustrating the relation sequence.]}
Construction Continued …

Now compose this again with \textsf{lcose}[0.5] and intersect with \texttt{ll}
Now compose this again with \textit{lc}lose[0.5] and intersect with \textit{ll}:
Construction Continued . . .

Now compose this again with $lclose[0.5]$ and intersect with $ll$:

In other words:

$$lclose[r/8] := (((lclose[r/2] \diamond rclose[0.5]) \cap sc) \diamond lclose[0.5]) \cap ll.$$
Now compose this again with $lclose[0.5]$ and intersect with $ll$:

In other words:

$$lclose[r/8] := (((lclose[r/2] \diamond rclose[0.5]) \cap sc) \diamond lclose[0.5]) \cap ll.$$  

So, we get indeed an infinite sequence of relations.
Consequences?

- We could define a **weaker** version of composition, namely, the strongest relation constructible as a union out of the 21 base relations that covers $r_1 \Diamond r_2$. 
Consequences?

- We could define a **weaker** version of composition, namely, the strongest relation constructible as a union out of the 21 base relations that covers $r_1 \diamond r_2$.

- This would lead to **correct** restrictions.
Consequences?

- We could define a **weaker** version of composition, namely, the strongest relation constructible as a union out of the 21 base relations that covers $r_1 \circ r_2$.

- This would lead to **correct** restrictions.

- However, repeated application of intersection, permutation, and **weak composition** does not lead to **4-consistency**!
Consequences?

• We could define a **weaker** version of composition, namely, the strongest relation constructible as a union out of the 21 base relations that covers $r_1 \diamond r_2$.

• This would lead to **correct** restrictions.

• However, repeated application of intersection, permutation, and **weak composition** does not lead to **4-consistency**!

$\sim$ So, we can forget about generalizing 4-consistency to **strong $n$-consistency**.
• We could define a **weaker** version of composition, namely, the strongest relation constructible as a union out of the 21 base relations that covers $r_1 \diamond r_2$.

• This would lead to **correct** restrictions.

• However, repeated application of intersection, permutation, and **weak composition** does not lead to **4-consistency**!

$\sim$ So, we can forget about generalizing 4-consistency to **strong $n$-consistency**.

$\rightarrow$ Computational complexity?
Fragments containing Only the Base Relations

- In all qualitative CSP calculi so far, satisfiability of CSP containing only the base relations and the universal relation could be decided in polynomial time.
Fragments containing Only the Base Relations

- In all qualitative CSP calculi so far, satisfiability of CSP containing only the base relations and the universal relation could be decided in polynomial time.

**Theorem.** Satisfiability of constraint systems over \{sf, ⊤\} (\(\top\) being the universal relation) is NP-hard.
Fragments containing Only the Base Relations

- In all qualitative CSP calculi so far, satisfiability of CSP containing only the base relations and the universal relation could be decided in polynomial time.

Theorem. Satisfiability of constraint systems over \{sf, \top\} (\top being the universal relation) is NP-hard.

Proof. Use reduction from BETWEENNESS
Fragments containing Only the Base Relations

- In all qualitative CSP calculi so far, satisfiability of CSP containing only the base relations and the universal relation could be decided in polynomial time.

**Theorem.** Satisfiability of constraint systems over \( \{sf, \top\} \) (\( \top \) being the universal relation) is NP-hard.

**Proof.** Use reduction from BETWEENNESS:

Given a finite set \( M \), a collection \( C \) of ordered triples \((a, b, c)\) of distinct elements from \( M \), is there a one-to-one function \( f : M \to \{1, 2, \ldots, |M|\} \) such that for each \((a, b, c) \in C\), we have either \( f(a) < f(b) < f(c) \) or \( f(c) < f(b) < f(a) \)?
Fragments containing Only the Base Relations

- In all qualitative CSP calculi so far, satisfiability of CSP containing only the base relations and the universal relation could be decided in polynomial time.

**Theorem.** Satisfiability of constraint systems over \( \{sf, \top\} \) (\( \top \) being the universal relation) is NP-hard.

**Proof.** Use reduction from BETWEENNESS:

Given a finite set \( M \), a collection \( C \) of ordered triples \((a, b, c)\) of distinct elements from \( M \), is there a one-to-one function \( f : M \to \{1, 2, \ldots, |M|\} \) such that for each \((a, b, c) \in C\), we have either \( f(a) < f(b) < f(c) \) or \( f(c) < f(b) < f(a) \)?

**Reduction:** Add \( sf(x, y, m) \) for each \( m \in M \) and two fixed \( x, y \notin M \). Then add \( sf(a, b, c) \) for each triple \((a, b, c) \in C\).
• What if all constraints are base relations?
Base Relations Only

- What if all constraints are base relations?
- Usually this gives enough information to decide satisfiability by running PC, but remember the 5-element system!
What if all constraints are base relations?

Usually this gives enough information to decide satisfiability by running PC, but remember the 5-element system!

For the double cross, satisfiability is NP-hard even if we only have base relations (which are not atomic!)
Base Relations Only

- What if all constraints are base relations?

- Usually this gives enough information to decide satisfiability by running PC, but remember the 5-element system!

- For the double cross, satisfiability is NP-hard even if we only have base relations (which are not atomic!)

- Idea: Use the freedom you get because the base relations are not atomic!
Base Relations Only

• What if all constraints are base relations?

• Usually this gives enough information to decide satisfiability by running PC, but remember the 5-element system!

• For the double cross, satisfiability is NP-hard even if we only have base relations (which are not atomic!)

• Idea: Use the freedom you get because the base relations are not atomic!

→ Decision method different from constraint propagation?
Decidability of the Double Cross

- Translate relations into inequalities over polynomials with integer coefficients (can be solved in PSPACE)
- Example: $rp(a, b, c)$
Decidability of the Double Cross

- Translate relations into inequalities over polynomials with integer coefficients (can be solved in PSPACE)
- **Example**: $rp(a, b, c)$
  - Let $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ be the coordinates of $a, b$ and $c$, respectively.
Decidability of the Double Cross

- Translate relations into **inequalities over polynomials with integer coefficients** (can be solved in **PSPACE**)

- **Example:** \( rp(a, b, c) \)
  
  - Let \((a_1, a_2), (b_1, b_2), (c_1, c_2)\) be the coordinates of \(a, b\) and \(c\), respectively.
  
  - The right angle can be expressed by stating that the scalar product of \(\overrightarrow{a, b}\) and \(\overrightarrow{b, c}\) is zero:
    \[
    (b_1 - a_1)(c_1 - b_1) + (b_2 - a_2)(c_2 - b_2) = 0
    \]
Decidability of the Double Cross

- Translate relations into **inequalities over polynomials with integer coefficients** (can be solved in **PSPACE**)

- **Example**: \( rp(a, b, c) \)
  - Let \((a_1, a_2), (b_1, b_2), (c_1, c_2)\) be the coordinates of \(a, b\) and \(c\), respectively.
  - The right angle can be expressed by stating that the scalar product of \(\overrightarrow{a, b}\) and \(\overrightarrow{b, c}\) is zero: \((b_1 - a_1)(c_1 - b_1) + (b_2 - a_2)(c_2 - b_2) = 0\)
  - Orientation is determined by: \((b_1 - a_1)(c_2 - a_2) - (b_2 - a_2)(c_1 - a_1) < 0\)
Decidability of the Double Cross

- Translate relations into inequalities over polynomials with integer coefficients (can be solved in PSPACE)

- **Example**: $rp(a, b, c)$
  - Let $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ be the coordinates of $a, b$ and $c$, respectively.
  - The right angle can be expressed by stating that the scalar product of $\overrightarrow{a,b}$ and $\overrightarrow{b,c}$ is zero: $(b_1 - a_1)(c_1 - b_1) + (b_2 - a_2)(c_2 - b_2) = 0$
  - Orientation is determined by: $(b_1 - a_1)(c_2 - a_2) - (b_2 - a_2)(c_1 - a_1) < 0$

$\Rightarrow$ Both constraints together ensure that the points can only be instantiated in a way satisfying the $rp$ constraint.
Decidability of the Double Cross

- Translate relations into inequalities over polynomials with integer coefficients (can be solved in PSPACE)

- **Example**: $rp(a, b, c)$
  - Let $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ be the coordinates of $a, b$ and $c$, respectively.
  - The right angle can be expressed by stating that the scalar product of $\overrightarrow{a, b}$ and $\overrightarrow{b, c}$ is zero: $(b_1 - a_1)(c_1 - b_1) + (b_2 - a_2)(c_2 - b_2) = 0$
  - Orientation is determined by: $(b_1 - a_1)(c_2 - a_2) - (b_2 - a_2)(c_1 - a_1) < 0$

$\sim$ Both constraints together ensure that the points can only be instantiated in a way satisfying the $rp$ constraint.

**Theorem.** Satisfiability of CSPs in the double cross calculus is in PSPACE.
Conclusions

- The double cross calculus can be used to support navigation in a qualitative way
Conclusions

• The double cross calculus can be used to support navigation in a qualitative way

• It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method
Conclusions

- The double cross calculus can be used to support navigation in a qualitative way.
- It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method.
- The double cross relation system has no finite refinement that is closed under composition, permutation, and intersection.
Conclusions

- The double cross calculus can be used to support navigation in a qualitative way.

- It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method.

- The double cross relation system has no finite refinement that is closed under composition, permutation, and intersection.

- Satisfiability is NP-hard, even if only base relations are used.
Conclusions

- The double cross calculus can be used to support navigation in a qualitative way.
- It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method.
- The double cross relation system has no finite refinement that is closed under composition, permutation, and intersection.
- Satisfiability is NP-hard, even if only base relations are used.
- The satisfiability problem is in PSPACE.
Conclusions

• The double cross calculus can be used to support navigation in a qualitative way

• It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method

• The double cross relation system has no finite refinement that is closed under composition, permutation, and intersection

• Satisfiability is NP-hard, even if only base relations are used

• The satisfiability problem is in PSPACE

⇒ Open: Tight bounds
Conclusions

- The double cross calculus can be used to support navigation in a qualitative way.

- It is a ternary calculus, so conversion and composition have to be generalized if we want to use (a generalization) of the path-consistency method.

- The double cross relation system has no finite refinement that is closed under composition, permutation, and intersection.

- Satisfiability is NP-hard, even if only base relations are used.

- The satisfiability problem is in PSPACE.

Open: Tight bounds.

Open: Relaxations or specializations that are easier.
Literature


