Imperial College London

جامعة الملك سعود
King Saud University

REDUCTION OF ILP SEARCH SPACE WITH BOTTOM-UP PROPOSITIONALISATION

Hadeel Al-Negheimish and Alessandra Russo

Imperial College London
King Saud University

halnegheimish@ksu.edu.sa

MOTIVATION

- A typical learning task takes explicit language bias as input
- Specification of language bias requires human intuition:
  - Too constrained: Risks not finding a solution
  - Too inclusive: Might make search intractable

MOTIVATION

- A typical learning task takes explicit language bias as input
- Specification of language bias requires human intuition:
  - Too constrained: Risks not finding a solution
  - Too inclusive: Might make search intractable

How can we restrict the search space in a more automated way?

RELATED WORK

- Extracting meta-knowledge to restrict search e.g. [McCreath and Sharma, 1995]
- ILP frameworks without mode declarations
  - FOIL [Quinlan, 1990] and Meta-Interpretive Learning [Muggleton, 2015]
- Propositionalisation e.g. [Lavrač et al., 1991]

McCreath, E., Sharma, A.:Extraction of meta-knowledge to restrict the hypothesis space for ILP systems. In: AI-CONFERENCE pp. 75–82. Citeseer (1995)

Quinlan, J.R.: Learning logical definitions from relations. Machine learning 5(3), 239–266 (1990)

Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Mach Learn pp. 1–25 (Mar 2015)

Lavrač, N., Džeroski, S., Grobelnik, M.: Learning nonrecursive definitions of relations with linus. In: Proceedings of the European Working Session on Learning on Machine Learning. pp. 265–281. EWSL-91, Springer-Verlag New York, Inc., New York, NY, USA (1991)

RELATED WORK

- Extracting meta-knowledge to restrict search e.g. [McCreath and Sharma, 1995]
- ILP frameworks without mode declarations
  - FOIL [Quinlan, 1990] and Meta-Interpretive Learning [Muggleton, 2015]
- Propositionalisation e.g. [Lavrač et al., 1991]

McCreath, E., Sharma, A.:Extraction of meta-knowledge to restrict the hypothesis space for ILP systems. In: AI-CONFERENCE pp. 75–82. Citeseer (1995)
Quinlan, J.R.: Learning logical definitions from relations. Machine learning 5(3), 239–266 (1990)
Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Mach Learn pp. 1–25 (Mar 2015)
Lavrač, N., Džeroski, S., Grobelnik, M.: Learning nonrecursive definitions of relations with linus. In: Proceedings of the European Working Session on Learning on Machine Learning. pp. 265–281. EWSL-91, Springer-Verlag New York, Inc., New York, NY, USA (1991)

LEARNING TASK

Let $\mathcal{L}_{BPILP}$ be a language and let T be the learning task of our approach, such that T= <B, E+, E-> is defined in $\mathcal{L}_{BPILP}$, where:

B Background knowledge, a stratified normal logic program, with no occurrence of the target predicate anywhere in the bodies of rules.

E+ Positive examples, a set of ground atoms of the same predicate with no functor symbols

E- Negative examples, also a set of ground atoms with no functor symbols of the same predicate in E+

LEARNING TASK

Let $\mathcal{L}^*_{BPILP}$ be an extended language defined as $\{\mathcal{L}_{BPILP} \cup \varphi\}$ where φ is an invented predicate that does not appear anywhere in $\mathcal{L}_{BPILP}$. A solution H is a (set of) normal clause(s) such that:

1. $H \in \mathcal{L}^*_{BPILP}$
2. $\forall\ e+ \in E+ : B \cup H \models e+$
3. $\forall\ e- \in E- : B \cup H \not\models e-$

OUR APPROACH

- We develop an algorithmic approach that interleaves incremental feature-construction and selecting useful features using:
  - **Bottom-up Propositionalisation**
  - **Extended Set Operations** -Takes into account *generality*

- It works on normal logic observational learning tasks

EXAMPLE

- **Background**

```
child(X,Y):- son(X,Y).
child(X,Y):- daughter(X,Y).
son(sam, sara).
daughter(alice, mary).
female(sara).
female(alice).
female(mary).
male(sam).
```

- **Positive Examples**

```
mother(sara, sam).
mother(mary, alice).
```

- **Negative Examples**

```
mother(alice, mary).
mother(sam, alice).
```

EXAMPLE

1. **Get model(B):**

```
child(sam,sara).
child(alice, mary).
son(sam, sara).
daughter(alice, mary).
female(sara).
female(alice).
female(mary).
male(sam).
```

2. **Find Target:**

The head of the main clause of the hypothesis is the least general-generalisation of the positive examples

lgg(`mother(sara, sam),` `mother(mary, alice)` **)= mother(X,Y)**

- With mother(X,Y) as target

1. Find mapping of example terms to head vars
2. Find relevant atoms in model(B) by term linking
3. Generalise links wrt $\mathcal{M}$(e)

	e_1 (sara, sam) ⊕	**e_2 (mary, alice) ⊕**	**e_3 (alice, mary) ⊖**	**e_4 (sam, alice) ⊖**
Mapping $\mathcal{M}$(e)	sara ↦ X sam ↦ Y	mary ↦ X alice ↦ Y	alice ↦ X mary ↦ Y	sam ↦ X alice ↦ Y
Links(e)				
Features(e)				

- With mother(X,Y) as target

1. Find mapping of example terms to head vars
2. Find relevant atoms in model(B) by term linking
3. Generalise links wrt $\mathscr{M}$(e)

	e_1 (sara, sam) ⊕	e_2 (mary, alice) ⊕	e_3 (alice, mary) ⊖	e_4 (sam, alice) ⊖
Mapping $\mathscr{M}$(e)	sara $\mapsto$ X sam $\mapsto$ Y	mary $\mapsto$ X alice $\mapsto$ Y	alice $\mapsto$ X mary $\mapsto$ Y	sam $\mapsto$ X alice $\mapsto$ Y
Links(e)	child(sam, sara). son(sam, sara). female(sara). male(sam).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(sam,sara). child(alice, mary). son(sam, sara). daughter(alice, mary). female(alice). male(sam).
Features(e)				

- With mother(X,Y) as target

1. Find mapping of example terms to head vars
2. Find relevant atoms in model(B) by term linking
3. Generalise links wrt $\mathcal{M}$(e)

	e_1 (sara, sam) ⊕	e_2 (mary, alice) ⊕	e_3 (alice, mary) ⊖	e_4 (sam, alice) ⊖
Mapping $\mathcal{M}$(e)	sara ↦ X sam ↦ Y	mary ↦ X alice ↦ Y	alice ↦ X mary ↦ Y	sam ↦ X alice ↦ Y
Links(e)	child(sam, sara). son(sam, sara). female(sara). male(sam).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(sam,sara). child(alice, mary). son(sam, sara). daughter(alice, mary). female(alice). male(sam).
Features(e)	child(Y, X). son(Y, X). female(X). male(Y).	child(Y, X). daughter(Y, X). female(Y). female(X).	child(X,Y). daughter(X,Y). female(X). female(Y).	child(X,SK). child(Y, SK). son(X, SK). daughter(Y, SK). female(Y). male(X).

- With mother(X,Y) as target

Select Features:

$\cap^*$ feats(e^+) $\forall e^+ \in E^+$

$\cup$ feats(e^-) * $\cup$ feats(e^+)

	e_1 (sara, sam) ⊕	e_2 (mary, alice) ⊕	e_3 (alice, mary) ⊖	e_4 (sam, alice) ⊖
Mapping $\mathcal{M}$(e)	sara ↦ X sam ↦ Y	mary ↦ X alice ↦ Y	alice ↦ X mary ↦ Y	sam ↦ X alice ↦ Y
Links(e)	child(sam, sara). son(sam, sara). female(sara). male(sam).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(alice, mary). daughter(alice,mary). female(alice). female(mary).	child(sam,sara). child(alice, mary). son(sam, sara). daughter(alice, mary). female(alice). male(sam).
Features(e)	child(Y, X). son(Y, X). female(X). male(Y).	child(Y, X). daughter(Y, X). female(Y). female(X).	child(X,Y). daughter(X,Y). female(X). female(Y).	child(X,SK). child(Y, SK). son(X, SK). daughter(Y, SK). female(Y). male(X).

Saturated Hypothesis:
mother(X,Y):- child (Y,X), female(X), not child(X,Y), not daughter(X,Y), not male(X).

Reduced Hypothesis:
mother(X,Y):- child (Y,X), female(X).

SUMMARY

- **Bottom-up Propositionalisation (*feature construction*)**
  - **Get mapping $\mathscr{M}$**
  - **Find links via terms**
  - **Generalise atoms in Links wrt $\mathscr{M}$**
- **Extended Set Operations (*feature selection*)**

- **If a simple solution is not found, process is repeated to expand skolem variables or invent predicates**

THEORETICAL PROPERTIES

- Stratification of B U H
- Soundness
- Feature minimality
- Existence of solution

EVALUATION

- **Animals Dataset**
  - Divided to 4 tasks, each with one of the class constants
  - Resulting Hypothesis:

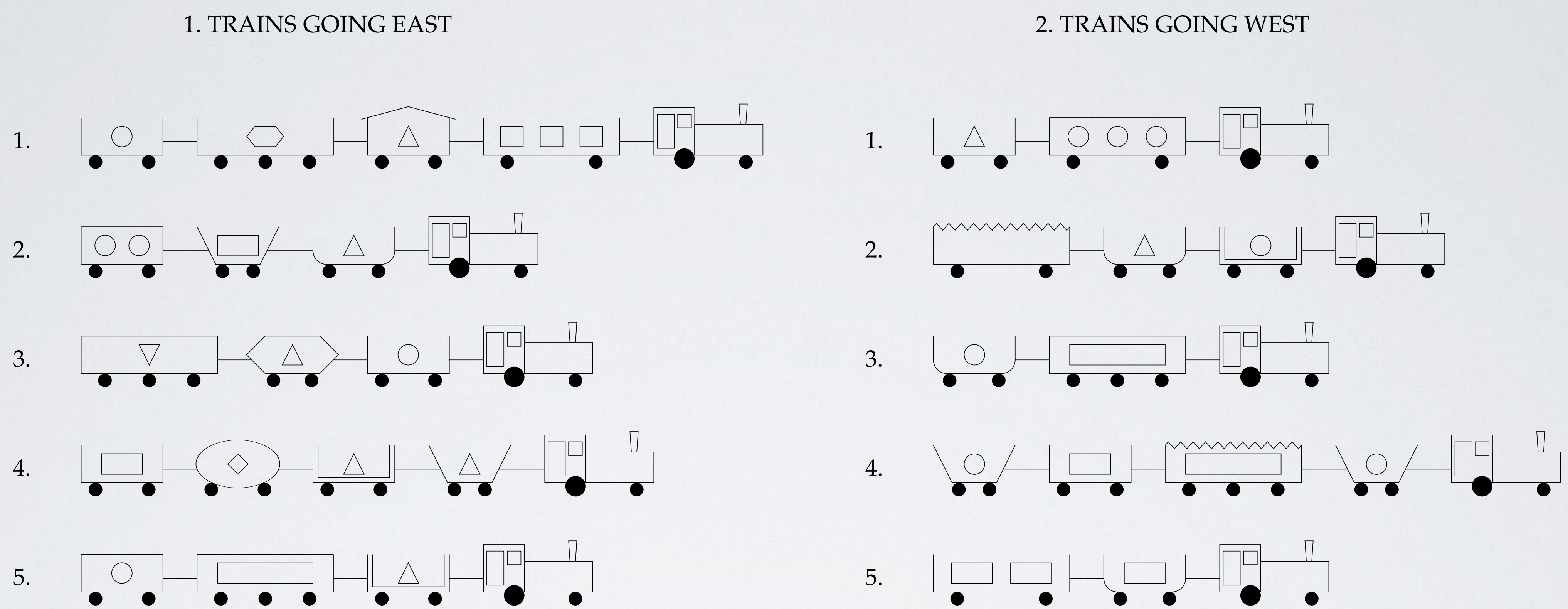
```
class(A, mammal):- has_milk(A).
class(A, fish):- has_gills(A).
class(A,bird):- has_covering(A, feathers).
class(A, reptile):- has_covering(A, scales),
not has_gills(A).
```

EVALUATION

- **Animals Dataset**
  - Divided to 4 tasks, each with one of the class constants
  - Resulting Hypothesis:

```
class(A, mammal):- has_milk(A).
class(A, fish):- has_gills(A).
class(A,bird):- has_covering(A, feathers).
class(A, reptile):- has_covering(A, scales),
not has_gills(A).
```

- # Eastbound Trains



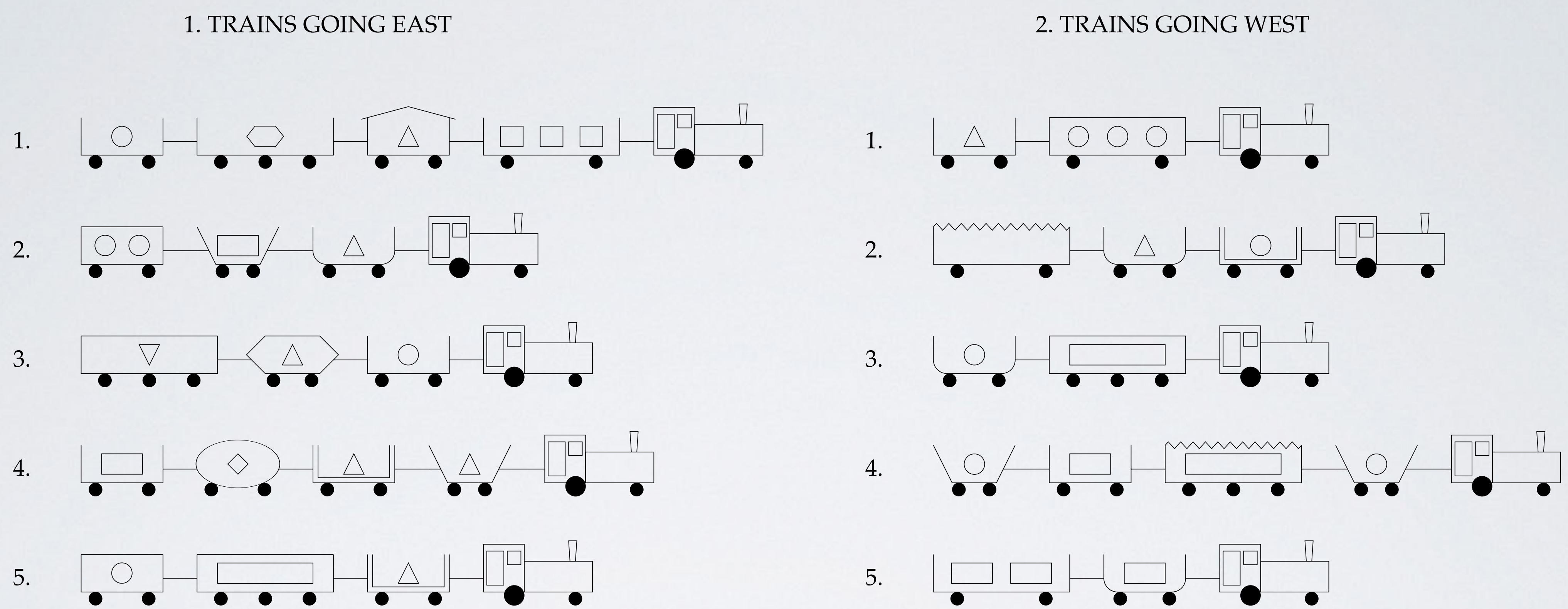
- with eastbound as E^+

```
eastbound(A):- closed(V1), shape(V1,V2), car(V1),
wheels(V1,V3), load(V1,V4,V5), short(V1), has_car(A,V1).
```

- with westbound as E^+

```
westbound(A):- not inv(A).
inv(A):- closed(V1), shape(V1,V2), car(V1),
wheels(V1,V3), load(V1,V4,V5), short(V1), has_car(A,V1).
```

• Eastbound Trains



- with eastbound as E^+

eastbound(A):- **closed(V1)**, shape(V1,V2), car(V1), wheels(V1,V3), load(V1,V4,V5), **short(V1)**, **has_car(A,V1)**.

- with westbound as E^+

westbound(A):- not **inv(A)**.
inv(A):- *closed(V1), shape(V1,V2), car(V1), wheels(V1,V3), load(V1,V4,V5), short(V1), has_car(A,V1).*

CONCLUSION

- We have introduced a method for bottom-up propositionalisation and used it for algorithmic learning *without* explicit language bias

CONCLUSION

- We have introduced a method for bottom-up propositionalisation and used it for algorithmic learning *without* language bias

Strengths

- It learns normal logic programs
- Supports predicate invention

CONCLUSION

- We have introduced a method for bottom-up propositionalisation and used it for algorithmic learning *without* language bias

Strengths

- It learns normal logic programs
- Supports predicate invention

Weaknesses

- Observational learning only
- Works best on single-clause hypothesis
- Does not work well with recursive definitions

EXTRA

$\cap^*$

- Let x, y be head variables
- Let $A = \{p(x, x), p(SK, y), q(x)\}$
  and $B = \{p(x, x), p(x, y), q(y)\}$
- $A \cap^* B =$

EXTRA

$\cap^*$

- Let x, y be head variables

- Let $A = \{\mathbf{p(x, x)}, p(SK, y), q(x)\}$
  and $B = \{\mathbf{p(x, x)}, p(x, y), q(y)\}$

- $A \cap^* B = \{\mathbf{p(x, x)}, ..\}$ since $p(x, x) \models p(x, x)$

EXTRA

$\cap^*$

- Let x, y be head variables

- Let A = {p(x, x), p(SK, y), q(x)}
  and B ={p(x, x), p(x, y), q(y)}

- $A \cap^* B = \{p(x, x), p(SK, y)\}$ since $p(SK, y) \models p(x, y)$

EXTRA

$\cap^*$

- Let x, y be head variables
- Let $A = \{p(x, x), p(SK, y), q(x)\}$ and $B = \{p(x, x), p(x, y), q(y)\}$
- $A \cap^* B = \{p(x, x), p(SK, y)\}$

$q(x) \not\models q(y)$

Since they are head variables they are treated as constants and cannot be unified to another value

EXTRA

Recursive Definitions

- At its current implementation, this approach does not work very well with recursive definitions
- This has to do with checking feature coverage, target predicate is treated the same as other features

EXTRA

Recursive Definitions

- Learnable Recursive Theory
  - vp(X,Y):- vp(X,Z), modif(Z,Y)

B	E^+	E^-
np(0,1). vp(1,2). modif(2,3). np(4,5). vp(5,6). modif(6,7).	vp(1,3). vp(5,7).	vp(1,7).

EXTRA

Recursive Definitions

- Unlearnable Recursive Theory
  - ancestor(X,Y):- parent(X,Z), ancestor(Z,Y)

B	E^+	E^-
parent(a,b). parent(b,c). parent(b,f). parent(f,g). parent(c,d). parent(d,e). ancestor(X,Y):- parent(X,Y).	ancestor(a,c). ancestor(a,d). ancestor(b,e). ancestor(a,e).	ancestor(d,a). ancestor(f,e).

EXTRA

Multiple Mappings

- Let target be p(X,Y)
- Let e = p(bob, bob)
- $\mathscr{M}$(e) = { bob $\mapsto$ {X,Y}}

How do we generalise links in this case?

A feature is generated for each mapping

EXTRA

Multiple Mappings

- Let target be p(X,Y)
- Let e = p(bob, bob)
- $\mathscr{M}$(e) = { bob $\mapsto$ {X,Y}}

If q(bob) is in Links(e) then both

q(X) and **q(Y)**

are generalisations with respect to $\mathscr{M}$(e)

Reduction In Feature Templates

EXTRA

- Let t=|head vars|
- Any atom can be generalised in $(t+1)^n$ ways where n is the arity of the predicate

Reduction In Feature Templates

EXTRA

- Example:

Let t= 2 (target = p(X,Y))

let r/2 be a predicate, it can be generalised as

• r(X,X)	• r(Y,X)	• r(SK,X)
• r(X,Y)	• r(Y,Y)	• r(SK,Y)
• r(X, SK)	• r(Y, SK)	• r(SK, SK)