Applications of inductive types in artificial intelligence and inductive reasoning

Ekaterina Komendantskaya

School of Computing, University of Dundee

Presentation at CiE'2010

Computational Logic in Neural Networks

Symbolic Logic as Deductive System

- Deduction in logic calculi;
- Logic programming;
- Higher-order proof assistants...

Sound symbolic methods we can trust

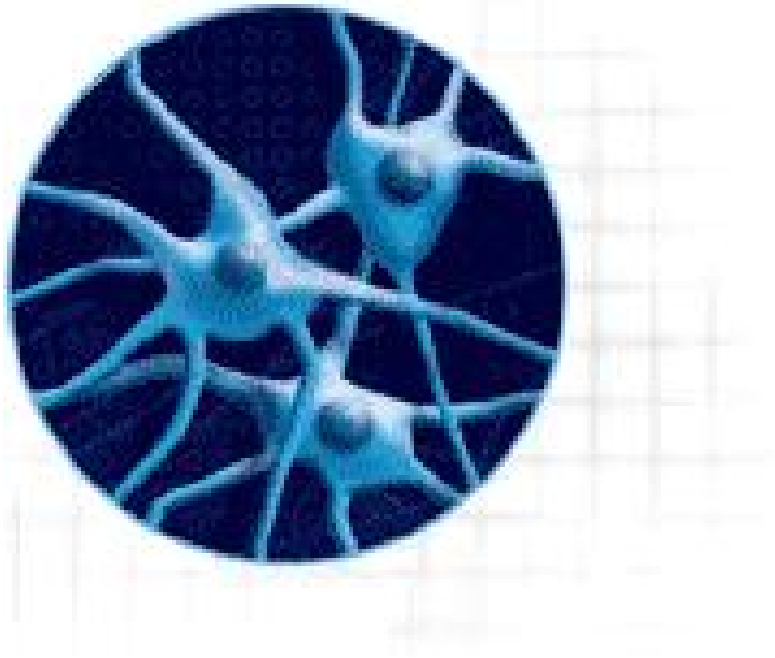
Computational Logic in Neural Networks

Symbolic Logic as Deductive System

- Deduction in logic calculi;
- Logic programming;
- Higher-order proof assistants...

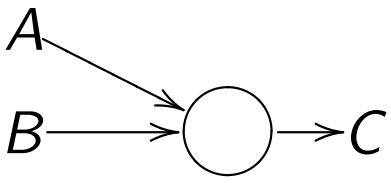
Sound symbolic methods we can trust

Neural Networks

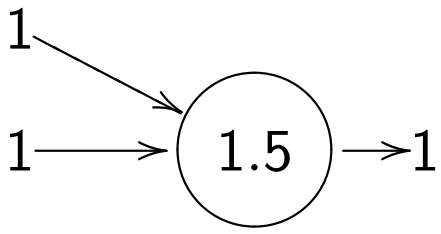


- spontaneous behavior;
- learning and adaptation;
- parallel computing.

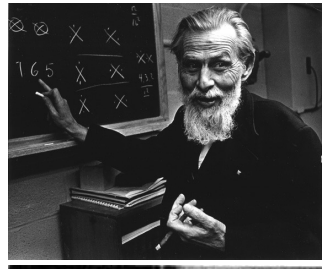
Boolean Networks of McCullogh and Pitts, 1943.

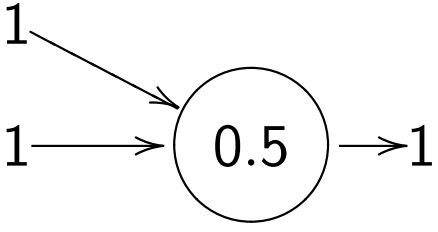


If A and B then C.

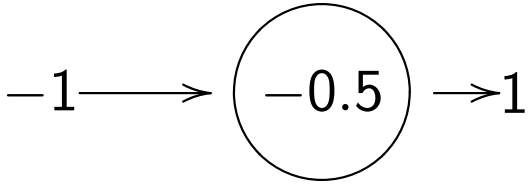


$(A = 1)$ and $(B = 1)$.





$(A = 1)$ or $(B = 1)$.



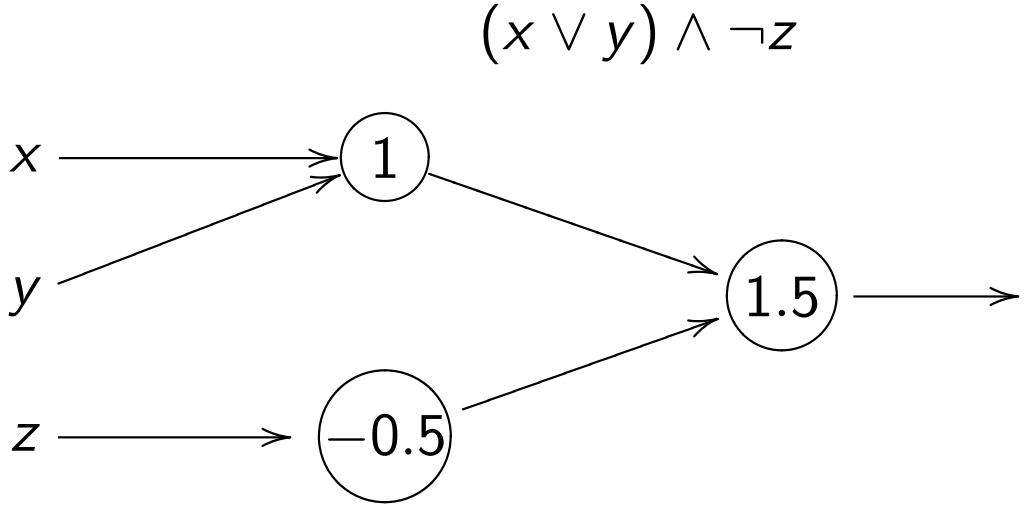
Not $(A = -1)$.

Neuro-symbolic architectures of other kinds based on the same methodology:

The approach of McCulloch and Pitts to processing truth values has dominated the area, and many modern neural network architectures consciously or unconsciously follow and develop this old method.

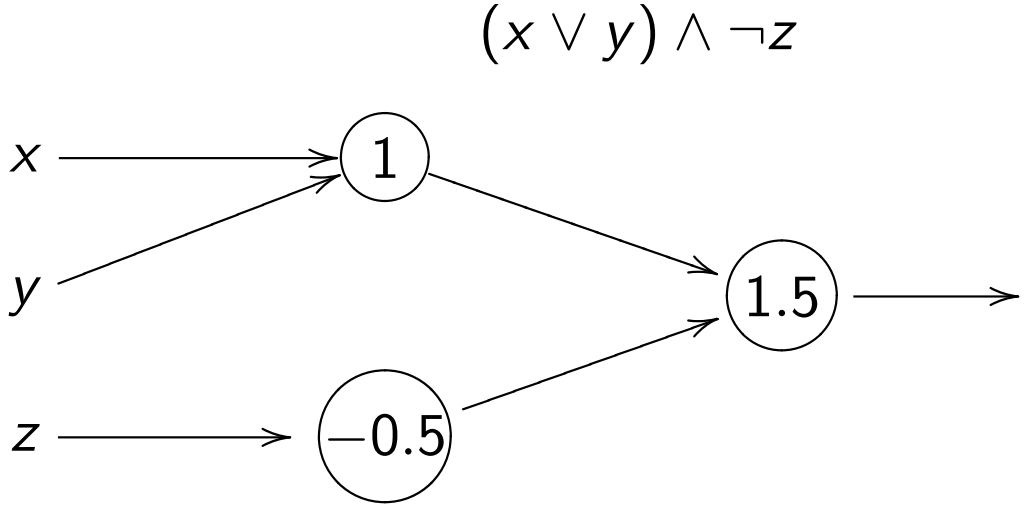
- Core Method: massively parallel way to compute minimal models of logic programs. [Holldobler et al, 1999 - 2009]
- Markov Logic and Markov networks: statistical AI and Machine learning implemented in NN. [Domingos et al., 2006-2009]
- Inductive Reasoning in Neural Networks [Broda, Garcez et al. 2002,2008]
- Fuzzy Logic Programming in Fuzzy Networks [Zadeh at al].

How do we know that they are correct?



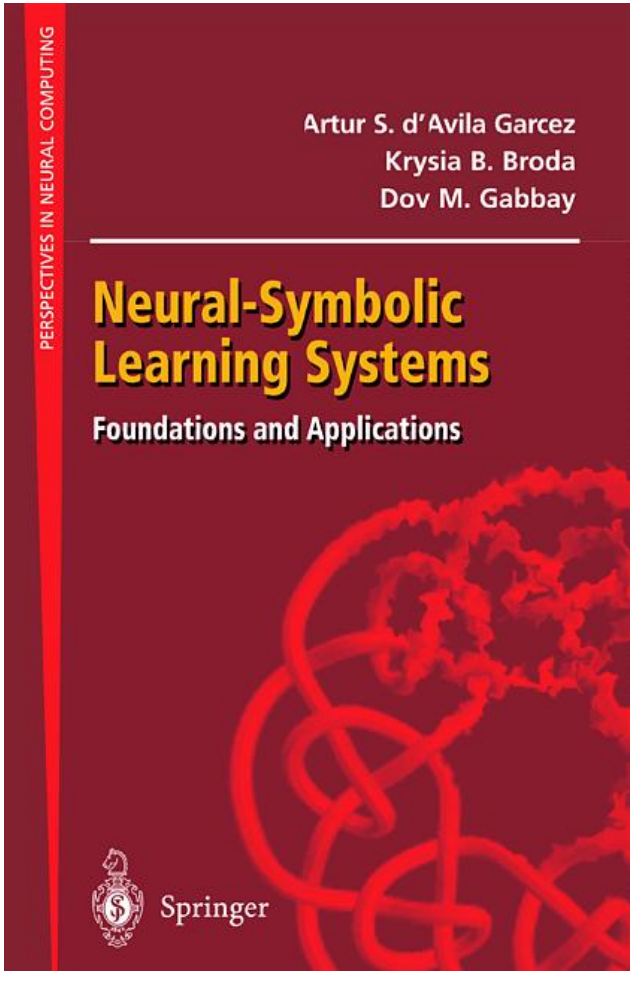
Such network would not distinguish "logical" data (values 0 and 1) from any other type of data, and would output the same result both for sound inputs like $x := 1, y := 1, z := 0$, and for non-logical values such as $x := 100.555, y := 200.3333\ldots, z := 0$. Imagine a user monitors the outputs of a big network, and sees outputs 1, standing for "true", whereas in reality the network is receiving some uncontrolled data.

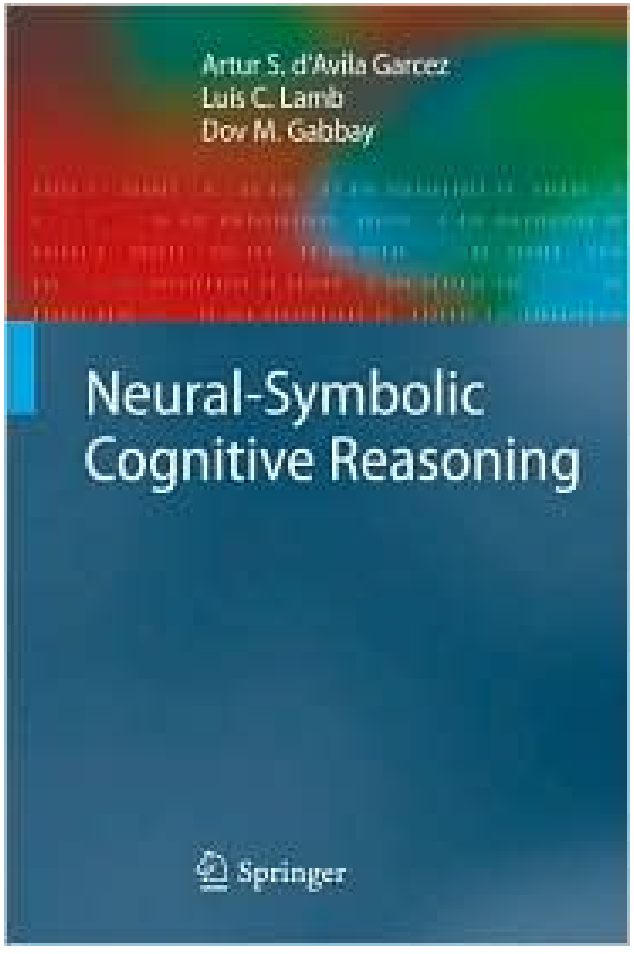
How do we know that they are correct?



Such network would not distinguish "logical" data (values 0 and 1) from any other type of data, and would output the same result both for sound inputs like $x := 1, y := 1, z := 0$, and for non-logical values such as $x := 100.555, y := 200.3333\ldots, z := 0$. Imagine a user monitors the outputs of a big network, and sees outputs 1, standing for "true", whereas in reality the network is receiving some uncontrolled data. The network gives correct answers on the condition that the input is well-typed.

Relational learning





Relational Reasoning and Learning.

In [Garcez et al, 2009], were built networks that can learn relations. E.g., given examples $Q(b, c) \rightarrow P(a, b)$ and $Q(d, e) \rightarrow P(c, d)$, they can infer a more general relation $Q(y, z) \rightarrow P(x, y)$.

Example

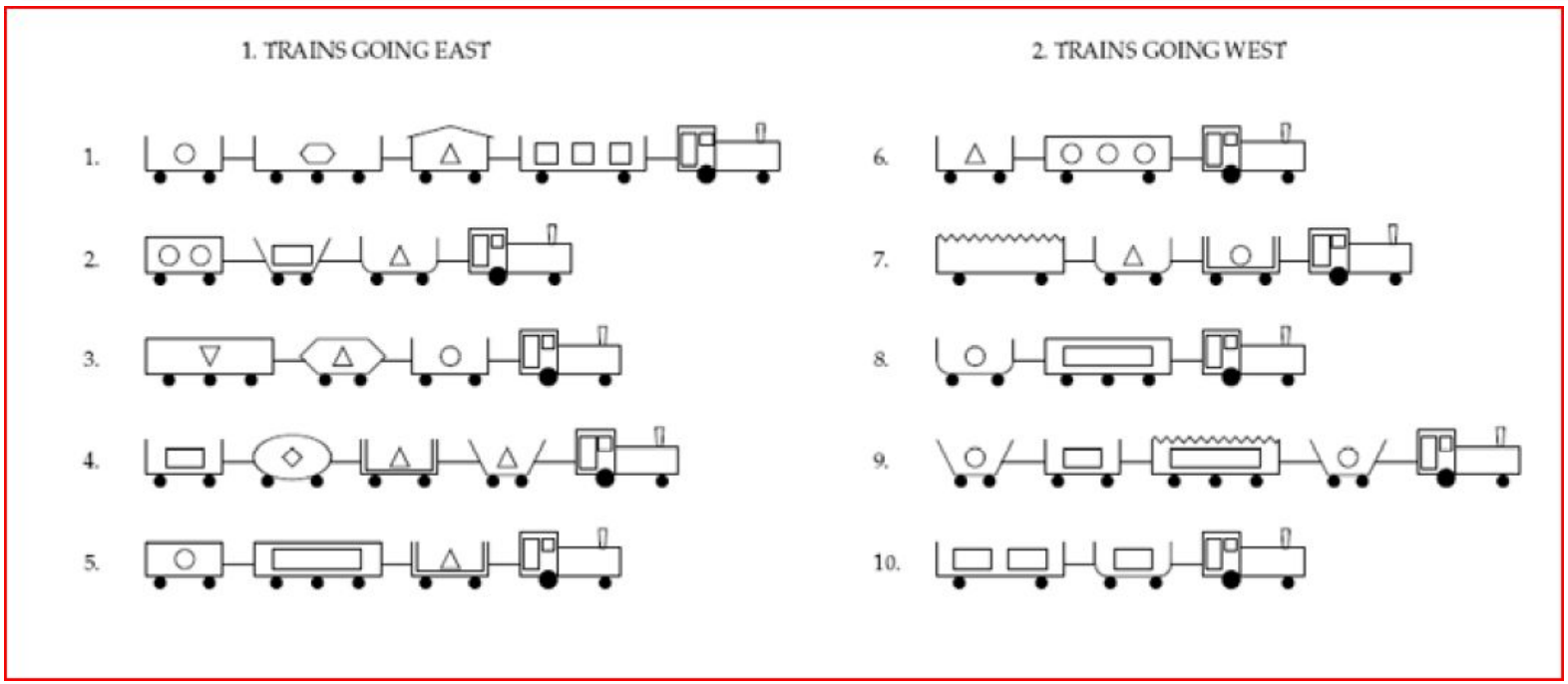
Learning a relation "grandparent" by examining families.
Classification of trains according to certain characteristics.

Relational Reasoning and Learning.

In [Garcez et al, 2009], were built networks that can learn relations. E.g., given examples $Q(b, c) \rightarrow P(a, b)$ and $Q(d, e) \rightarrow P(c, d)$, they can infer a more general relation $Q(y, z) \rightarrow P(x, y)$.

Example

Learning a relation "grandparent" by examining families.
Classification of trains according to certain characteristics.



Problems with this method

Such relational learning works as long as input data is well-typed.

"Well-typed" means that only related people, and not any other objects, are given to the network that learns relation "grandparent". And there are only trains of particular, known in advance, configuration, that are considered by the network that classifies trains.

This means that users have to make the preliminary classification and filtering of data before it is given to such networks; and NNs would not be able to warn the users if the data are ill-typed :-(.

Generally, as it turns out, typing is important for correct reasoning.

One can generalise from "This dog has four legs, and hence it can run" to "Everything that has four legs can run". However, we know that there are some objects, such as chairs, that have four legs but do not move. Hence we (often unconsciously) use typing in such cases, e.g., apply the generalisation only to all animals.

Solutions: K.K., K. Broda, A.Garcez

Solution

As an alternative to the manual pre-processing of data, we propose neural networks that can do the same automatically. We use *neural networks called type recognisers*; and implement such networks to ensure the correctness of neural computations; both for classical cases (McCulloch & Pitts) and for the relational reasoning and learning.

Solutions: K.K., K. Broda, A.Garcez

Solution

As an alternative to the manual pre-processing of data, we propose neural networks that can do the same automatically. We use *neural networks called type recognisers*; and implement such networks to ensure the correctness of neural computations; both for classical cases (McCulloch & Pitts) and for the relational reasoning and learning.

The solution involves techniques like pattern-matching, inductive type definitions, etc. that are used in functional programming, type theory, and interactive theorem provers!

The main result

First ever method of using Types for ensuring the correctness of Neural or Neuro-Symbolic computations.

Theorem

For any type A, given an expression E presented in a form of a numerical vector, we can construct a neural network that recognises whether E is of type A.

Such networks are called Type recognisers, and for each given type A, the network that recognises expressions of type A is called an A-recogniser. This construction covers simple types, such as Bool, as well as more complex inductive types, such as natural numbers, lists; or even dependent inductive types, such as lists of natural numbers.

Some examples of inductive types

Primitive:

```
Inductive bool : Type := | t : bool
 | f : bool.
```

Some examples of inductive types

Primitive:

```
Inductive bool :  Type :=  | t :  bool
 | f :  bool.
```

Recursive:

```
Inductive nat :  Set :=
 | O : nat
 | S : nat -> nat.
```

Typical element of the set `SSSO`.

Some examples of inductive types

Primitive:

```
Inductive bool :  Type :=  | t :  bool
 | f :  bool.
```

Recursive:

```
Inductive nat :  Set :=
 | O : nat
 | S : nat -> nat.
```

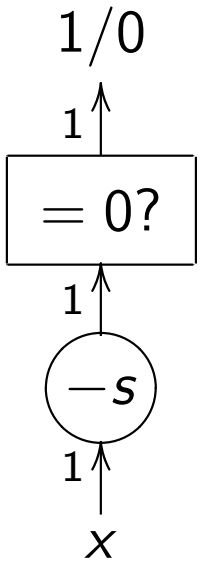
Typical element of the set `SSSO`.

Dependent:

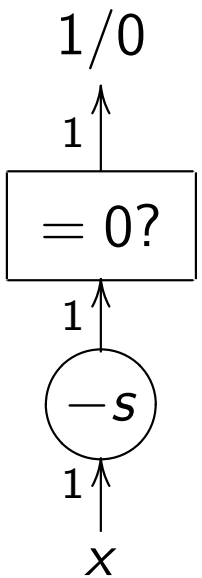
```
Inductive list nat :  Set :=
 | nil :  list
 | cons :  nat -> list -> list.
```

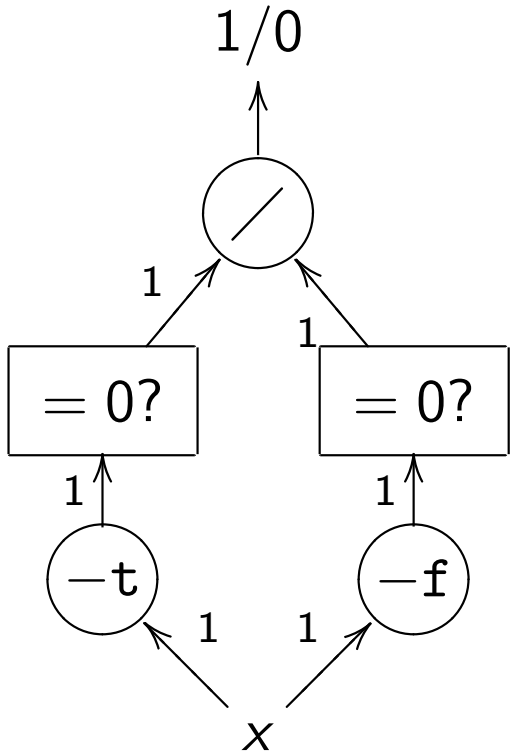
Typical element of the set is `O::SO::SSSO::O::nil` also written `cons O cons SO cons SSSO cons O cons nil`.

"Atomic" symbol recognisers

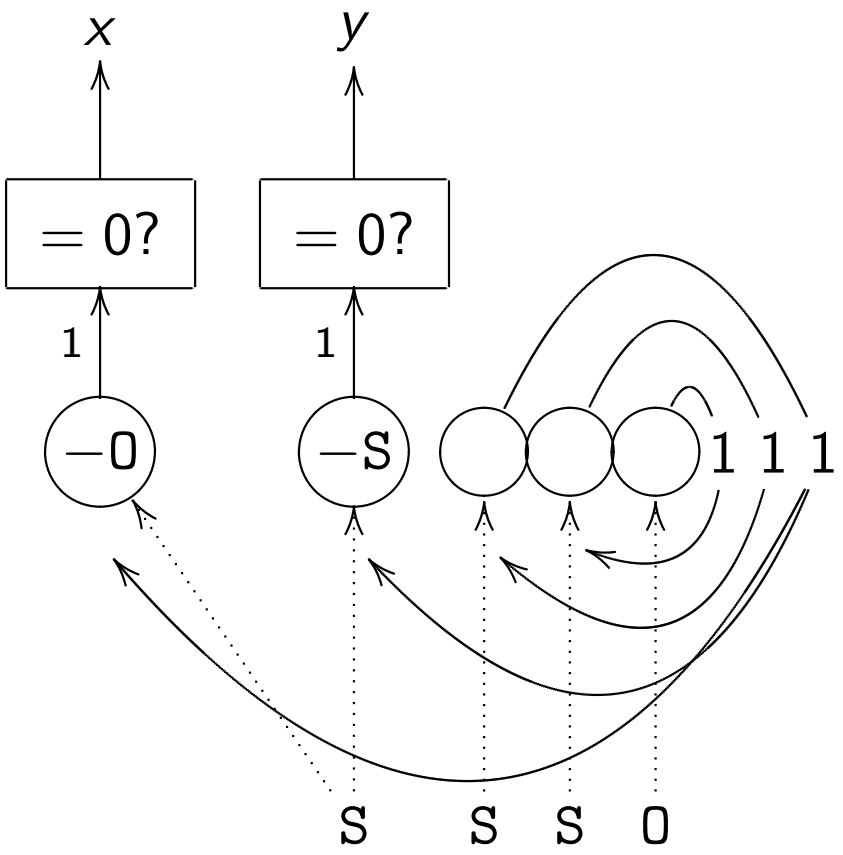


Inductive recogniser for bool



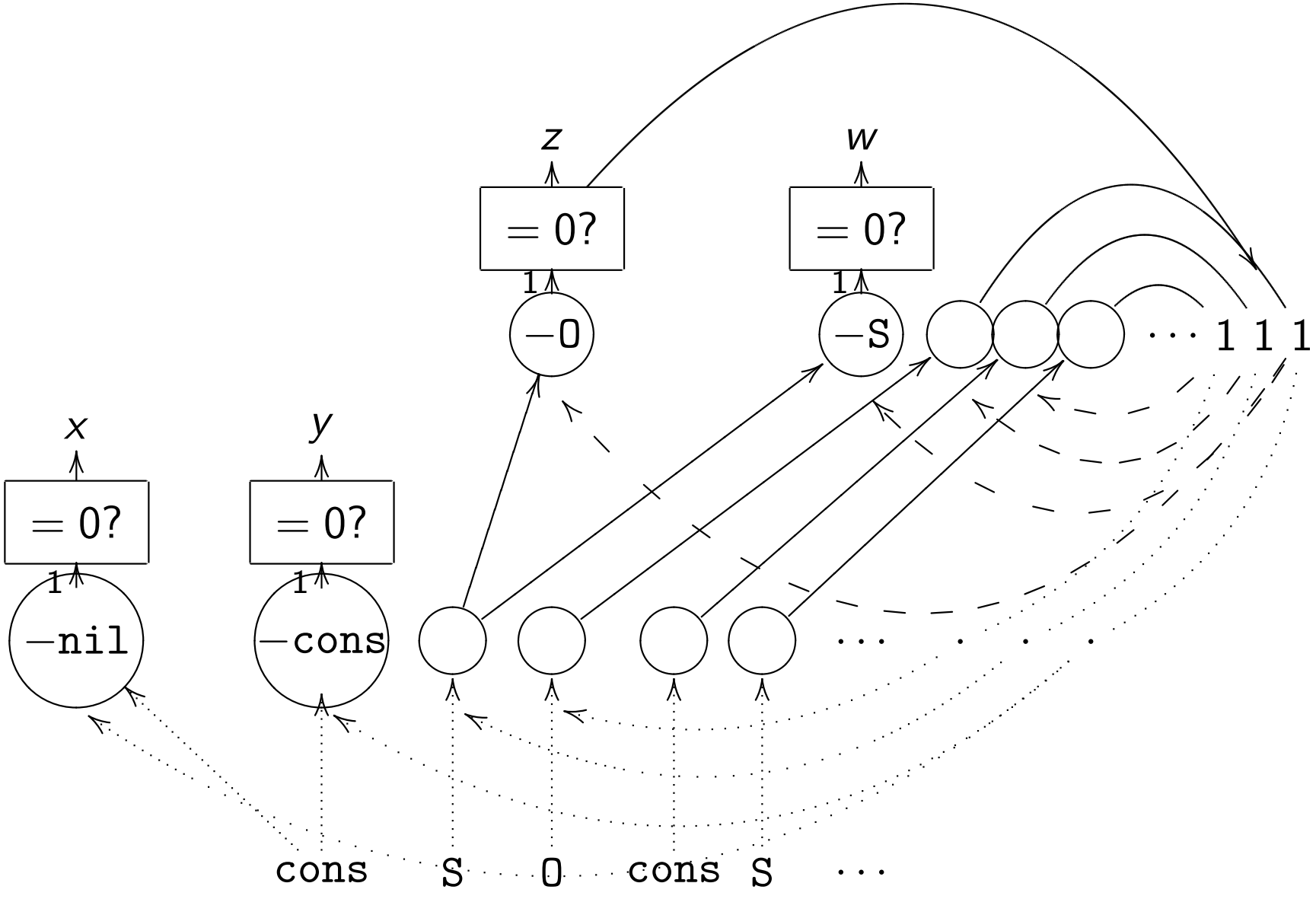


Inductive recogniser for `nat`

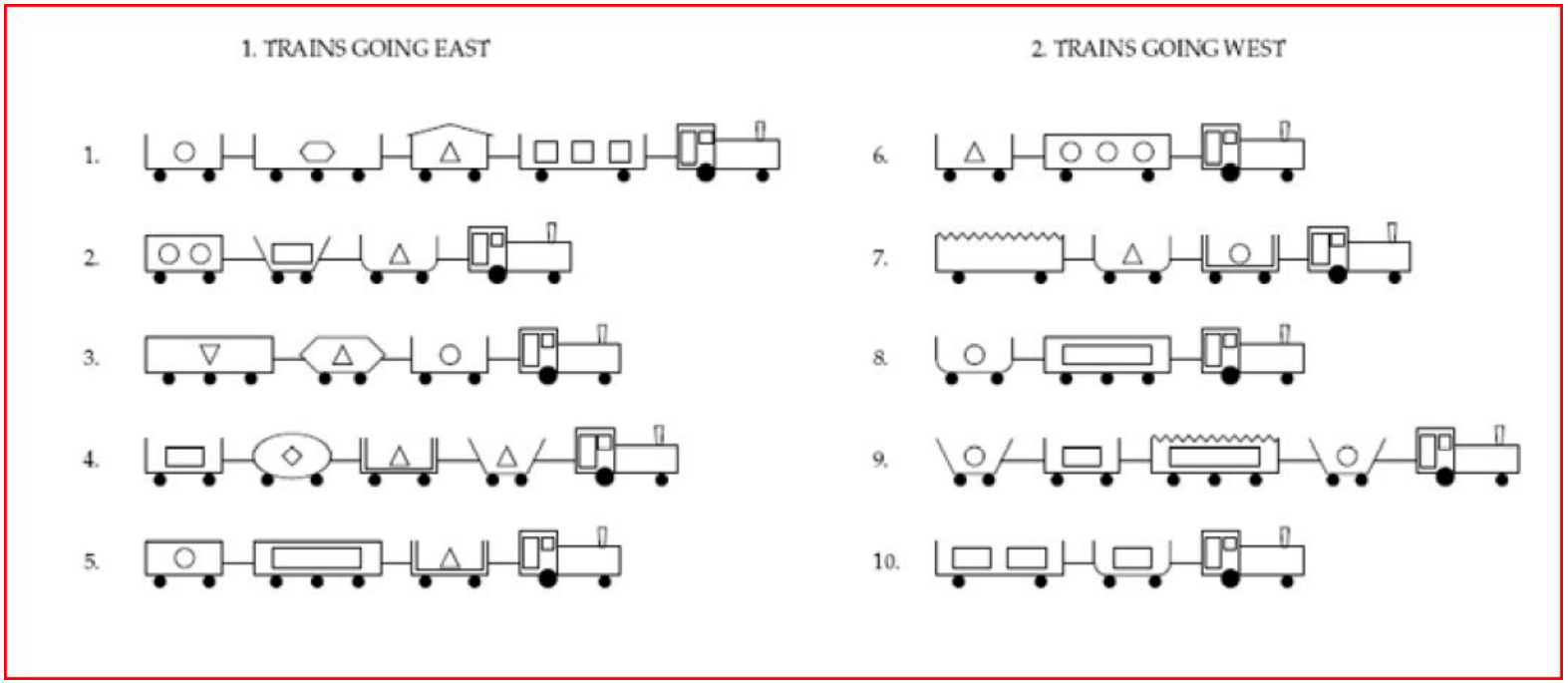


x	y	
1	0	- success
0	1	- working
1	1	- impossible
0	0	- failure

Inductive recogniser for `list nat`



Example with trains




```
Inductive shape :  Type :=
 | oval :  shape
 | triangle :  shape.
```

```
Inductive direction :  Set :=
 | west :  direction
 | east :  direction.
```

Inductive types in Neuro-symbolic networks

Inductive types can naturally be represented in neural networks:

- For finite sets, we use feed-forward networks
- For infinite sets defined recursively we use recursive connections in networks
  - Inductive types are closely related to recursive structures that arise in Neural networks;
- The networks can be integrated into big Neuro-symbolic systems to type-check inputs/outputs;
- The networks can be used for inductive generalisations and analogy.

Applications and Future work

- Types and type-theoretic approach has a big future in AI: be it inductive reasoning, learning techniques, or neuro-symbolic integration.
  - Our next step is to merge the networks with learning algorithms.
- They can be used for Logic Programming and Symbolic AI
  - Logic programs as inductive definitions paradigm $\longrightarrow$ ICANN paper.
- Inductive types should be used to ensure safety and security of Neuro-Symbolic networks;

Thank you!

Questions?