

Translating Logic into Neural Networks

School of Computer Science
University of St. Andrews

March 18th, 2009

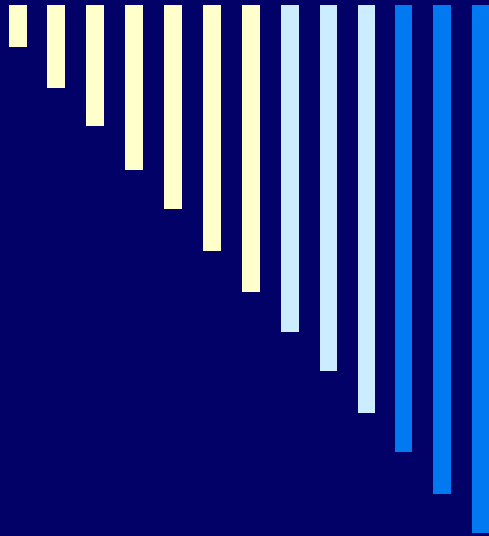
Kai-Uwe Kühnberger (joined work with Helmar Gust & Peter Geibel)
Institute of Cognitive Science (IKW)





Overview

- Motivation
- Logic and Models
- Abstract Models
- Translating Axioms to Arrows in a Topos
- Learning the Composition Operator of a Topos
- Conclusions



Motivation

Classical Problems of Neural-Symbolic Integration



Neural-Symbolic Integration

- The symbolic-subsymbolic distinction
 - There is an obvious tension between symbolic and subsymbolic representations.

	Symbolic Approaches	Subsymbolic Approaches
Methods	Mainly logical and / or algebraic	Mainly analytic
Strengths	Productivity, recursion, compositionality	Robustness, learning, parsimony, adaptivity
Weaknesses	Consistency constraints, lower cognitive abilities	Opaqueness, higher cognitive abilities
Applications	Reasoning, problem solving, planning etc.	Learning, motor control, vision etc.
Relation to Neurobiology	Not biologically inspired	Biologically inspired
Other Features	Crisp	Fuzzy



Problems of Integration

- A major problem of the modeling of predicate logic theories with connectionist systems is the treatment of variables (variable binding problem).
 - Blue small cars: `car(X), color(X,blue), size(X,small)`.
 - A neural representation needs to have a method of binding, for example, the color `blue` with the size `small`.
 - Another problem: How can values of variables be represented which can be changed during reasoning processes?
 - Etc.
- There are several approaches to address the variable binding problem in connectionist systems.
 - But there is no uncontroversial solution yet.



Examples of Proposed Solutions

- Examples of solving the variable binding problem:
 - Sign propagation: Lange & Dyer (1989), Sun & Waltz (1991)
 - Pre-wired static representation
 - Dynamical localist representations: Barnden (1989)
 - Hard-wired network is used to manage the array of registers (working memory).
 - Distributed representations used for resolution: Browne & Sun (1999)
 - The symbolic representations of pre-unification term pairs and unification results are transformed into distributed representations on the hidden layers of autoassociators.
 - Tensor product representations: Smolensky, 1990
 - Variable-value pairs are represented by the tensor product of the vector representing the variable and the vector representing the value.
 - Exponentially increasing number of elements to represent bindings.



Examples of Proposed Solutions

- Holographically reduced representations.
 - Plate, Gayler, Levy, Kanerva, Arathorn and others.
 - Avoid the exponential explosion of the vector space dimensionality.
- Modeling the semantics of logic programs with neural networks (“core method”)
 - Probably the best-known approach.
 - Classical papers by Hitzler, Hölldobler, Bader, Witzel and other.
 - Approximation of a model for logic programs using recurrent networks.
- Applications of neural-symbolic representations for non-classical logics
 - Work by Artur D’Avila Garcez, Dov Gabbay, Ekaterina Komendantskaya and others.



Problems of Integration

- More abstractly there are problems of learning heterogeneous data structures with connectionist systems.
 - Due to the fact that the model-theoretic interpretation of logical formulas is heterogeneous, it is not obvious how to implement interpretation functions.
- A further problem is the compositional nature of logic theories.
- There may be a confusion what it means to learn a logical theory.
 - In analogy to the lively discussion of the possibility of language learning, learning logical theories seems to be a natural question.



Logic and Models

Some Well-Known Things



Logic: Syntax and Semantics

- The classical way of doing logic is to distinguish syntax and semantics of logical theories.
- Predicate logic

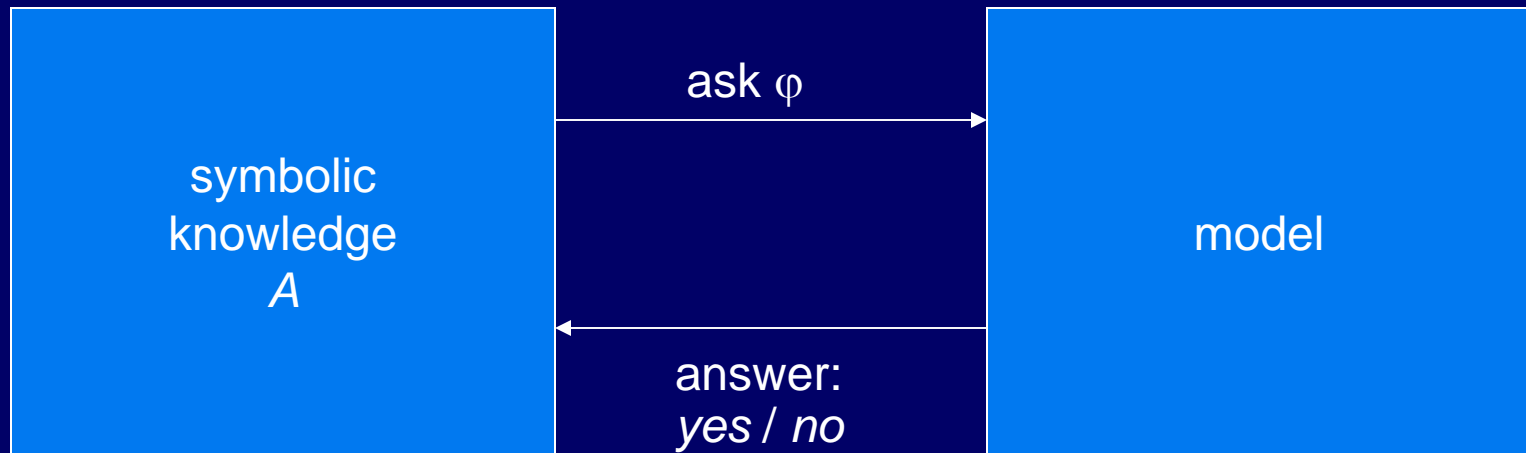
	Syntax	Semantics
Constants	a, b, \dots	$[[a]] \in \mathcal{U}$
Variables	x, y, \dots	$[[x]] \in \mathcal{U}$
Functions	f	$[[f]] : \mathcal{U} \times \dots \times \mathcal{U} \rightarrow \mathcal{U}$
Terms	$f(t_1, \dots, t_n)$	$[[f(t_1, \dots, t_n)]] = [[f]]([[t_1]], \dots, [[t_n]])$
Predicates	p	$[[p]] \subseteq \mathcal{U} \times \dots \times \mathcal{U}$
Atomic formulas	$p(t_1, \dots, t_n)$	$([[t_1]], \dots, [[t_n]]) \in [[p]]$
Formulas	$\neg A, A \vee B, A \wedge B$	$[[\neg A]] = \text{true}$ iff $[[A]] = \text{false}, \dots$
	$\forall xA, \exists xA$	$[[\forall x A]] = \text{true}$ iff $[[A]]_{x=e} = \text{true}$ for all $e \in \mathcal{U}$



Logic: Syntax and Semantics

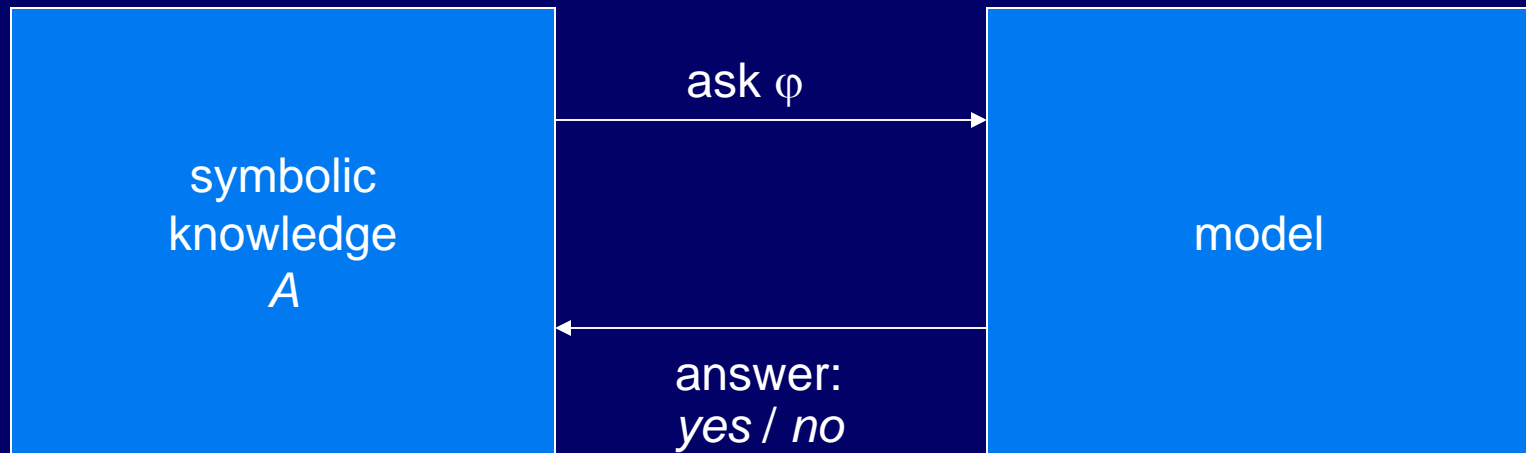
- Model
 - A structure $\mathcal{M} = \langle \mathcal{U}, [[\cdot]] \rangle$ with $[[\varphi]] = \text{true}$ is called a model for φ ($\mathcal{M} \models \varphi$).
- Validity
 - If every structure \mathcal{M} is a model for φ we call φ valid ($\models \varphi$).
- Satisfiability
 - If there exists a model \mathcal{M} for φ we call φ satisfiable.
- Logical Consequence
 - A formula φ is a logical consequence (or a logical entailment) of $A = \{A_1, \dots, A_n\}$, if each model for A is also a model for φ ($A \models \varphi$)
- Fundamental theorem of FOL
 - $A \models \varphi$ iff $A \vdash \varphi$.

Abstract Models



if $A \models \varphi$ then $\text{answer}(\text{ask } \varphi) = \text{yes}$
if $A \models \neg\varphi$ then $\text{answer}(\text{ask } \varphi) = \text{no}$
else $\text{answer}(\text{ask } \varphi) = \text{yes or no}$

Abstract Models



if $\text{answer}(\text{ask } \varphi) = \text{yes}$ then $\text{not}[A \models \neg\varphi]$
if $\text{answer}(\text{ask } \varphi) = \text{no}$ then $\text{not}[A \models \varphi]$



What is the 'meaning' of these Axioms?

- $\forall x: C(x,x)$
- $\forall x,y: C(x,y) \rightarrow C(y,x)$
- $\forall x,y: P(x,y) \leftrightarrow \forall z: (C(z,x) \rightarrow C(z,y))$
- $\forall x,y: O(x,y) \leftrightarrow \exists z: (P(z,x) \wedge P(z,y))$
- $\forall x,y: DC(x,y) \leftrightarrow \neg C(x,y)$
- $\forall x,y: EC(x,y) \leftrightarrow C(x,y) \wedge \neg O(x,y)$
- $\forall x,y: PO(x,y) \leftrightarrow O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$
- $\forall x,y: EQ(x,y) \leftrightarrow P(x,y) \wedge P(y,x)$
- $\forall x,y: PP(x,y) \leftrightarrow P(x,y) \wedge \neg P(y,x)$
- $\forall x,y: TPP(x,y) \leftrightarrow PP(x,y) \wedge \exists z(EC(z,x) \wedge EC(z,y))$
- $\forall x,y: TPPI(x,y) \leftrightarrow PP(y,x) \wedge \exists z(EC(z,y) \wedge EC(z,x))$
- $\forall x,y: NTPP(x,y) \leftrightarrow PP(x,y) \wedge \neg \exists z(EC(z,x) \wedge EC(z,y))$
- $\forall x,y: NTPPI(x,y) \leftrightarrow PP(y,x) \wedge \neg \exists z(EC(z,y) \wedge EC(z,x))$



Is This a Theorem?

□ $\forall x,y,z: \text{NTPP}(x,y) \wedge \text{NTPP}(y,z) \rightarrow \text{NTPP}(x,z)$

□ Easy to see if we look at models!

Relations of Regions of the RCC-8

(a canonical model: n -dimensional closed discs)

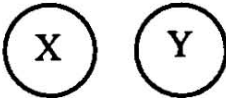
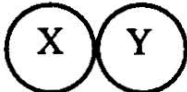
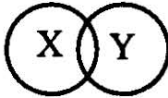
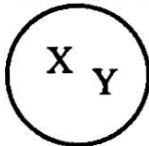
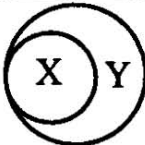
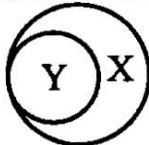
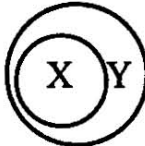
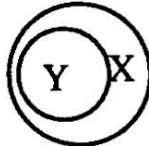
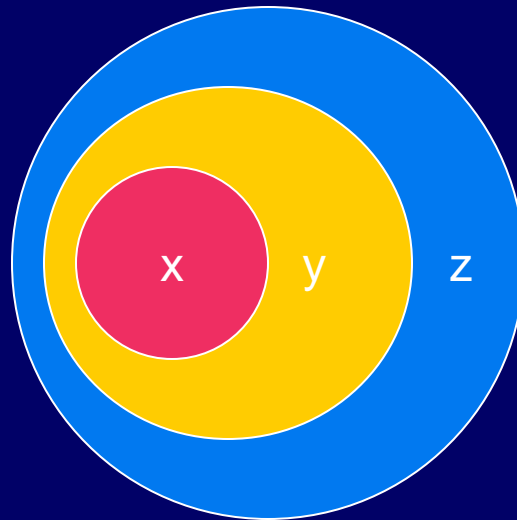
DC(X, Y) DisConnected		EC(X, Y) Externally Connected	
PO(X, Y) Partially Overlapping		EQ(X, Y) Equal	
TPP(X, Y) Tangential Proper Part		TPPI(X, Y) Tangential Proper Part Inverse	
NTPP(X, Y) Non-Tangential Proper Part		NTPPI(X, Y) Non-Tangential Proper Part Inverse	

Tabelle 10.2: Die Relationen des Region Connection Calculus

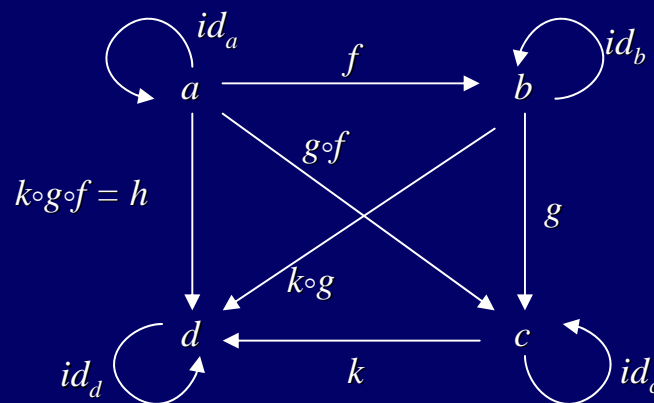
Is This a Theorem?

□ $\forall x,y,z: \text{NTPP}(x,y) \wedge \text{NTPP}(y,z) \rightarrow \text{NTPP}(x,z)$



The Definition of a Category

- A category \mathbf{C} is defined as a 6-tuple $\mathbf{C} = \langle \text{Obj}_{\mathbf{C}}, \text{Ar}_{\mathbf{C}}, \circ_{\mathbf{C}}, \text{dom}, \text{codom}, \text{id} \rangle$
 - $\text{Obj}_{\mathbf{C}}$: class of objects
 - $\text{Ar}_{\mathbf{C}}$: class of arrows (between objects)
 - $\circ_{\mathbf{C}}$: concatenation of arrows (associative)
 - dom, codom : defined on arrows (domain and codomain)
 - id : for each object $b \in \text{Obj}_{\mathbf{C}}$: $\text{id}_b: b \rightarrow b$, such that for each $f: a \rightarrow b$ and $g: b \rightarrow c$, it holds: $\text{id}_b \circ f = f$ and $g \circ \text{id}_b = g$





The Idea of a Topos

- An example of a topos is the category **SET**
 - Objects are sets, arrows are set-theoretic functions
- A topos is a special type of category
 - It has a final object “!”
 - For every $c \in \text{Obj}_C$ there exists a unique $!_c: c \rightarrow !$
 - For sets the one-element set $\{*\}$ is final
 - Products (and coproducts)
 - $a \leftarrow a \times b \rightarrow b$ with $a \times b$ Cartesian product and the corresponding projections
 - Limits and colimits
 - In particular, the generalization of products called a pullback is important
 - Exponents
 - For all $a \rightarrow c^b$ there is a unique $a \times b \rightarrow c$
 - A subobject classifier
 - $\text{true}: ! \rightarrow \Omega$ where Ω is a truth value object (in **SET** a two-element set)
 - Intuition: the subobject classifier allows the characterization of subobjects by predicates (or characteristic functions)
 - Subsets in **SET** can be represented by characteristic functions



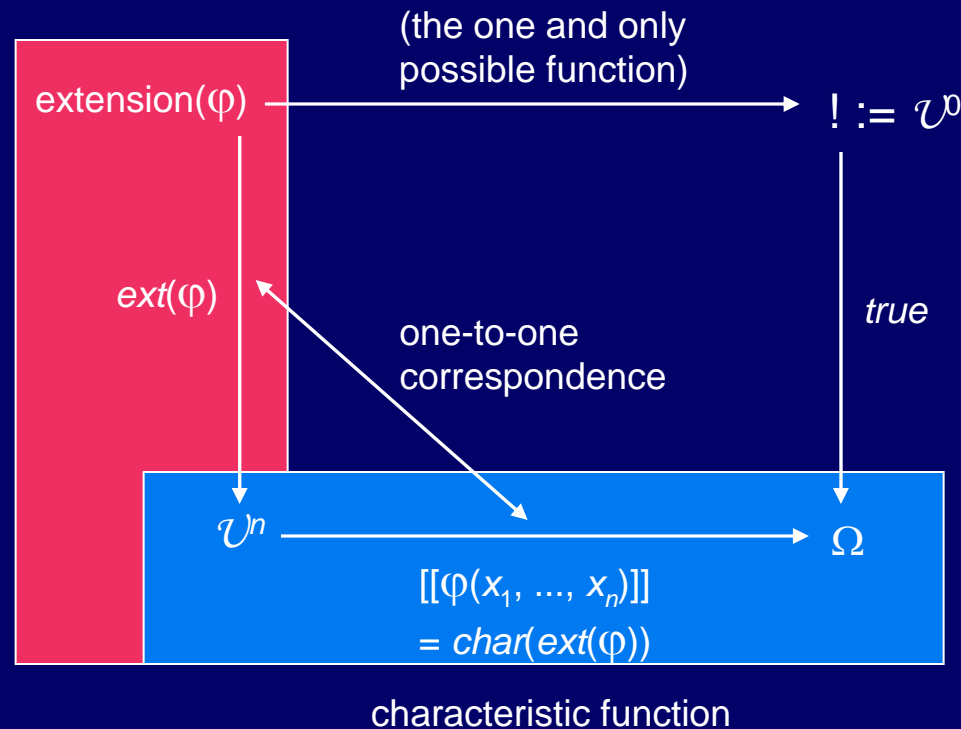
Abstract Models

- What do we need for an interpretation?
 - A universe \mathcal{U} (if we have a sorted logic, a universe for every sort)
 - Constants c : 0-arity functions
 $[[c]] : \mathcal{U}^0 \rightarrow \mathcal{U}$
 - Function symbols f : Products of the universe + functions
 $[[f]] : \mathcal{U}^n \rightarrow \mathcal{U}$
 - Relation symbols p :
 - Subsets of (products of) the universe
 - Truth-values: $\Omega = \{0, 1\}$
 - Functions from products of the universe to truth-values
 $[[p(x_1, \dots, x_n)]] : \mathcal{U}^n \rightarrow \Omega$
 - $[[true]]$, $[[false]]$: $\mathcal{U}^0 \rightarrow \Omega$

Abstract Models

- Extensions of predicates / generalization of the subset concept

For each monic arrow $ext(\varphi)$ there is a unique arrow $char(ext(\varphi))$ such that the diagram is a pullback.





Abstract Models

- What do we need for an interpretation (cont.)
 - Quantifiers ($Q \in \{\forall, \exists\}$):
 - A method for handling variables: currying

$$Qy_1, \dots, y_k: \varphi(x_1, \dots, x_n, y_1, \dots, y_k) = \psi(x_1, \dots, x_n)$$

- $[[\varphi(x_1, \dots, x_n, y_1, \dots, y_k)]]: \mathcal{U}^{n+k} \rightarrow \Omega$

- $[[Q y_1, \dots, y_k: \varphi(x_1, \dots, x_n, y_1, \dots, y_k)]]: \mathcal{U}^n \rightarrow \Omega$

- $[[Qy_1, \dots, y_k]]: \Omega^{U^{n+k}} \rightarrow \Omega^{U^n}$

- (There is a one-to-one mapping)

$$U^n \rightarrow \Omega \Leftrightarrow U^0 = \{()\} = ! \rightarrow \Omega^{U^n}$$



Abstract Models

- What do we get:
 - All interpretations of expressions or terms are functions between sets.
 - We never refer to the inner structure of functions and sets (i.e. we never refer to elements).
 - The only operation we use on functions is composition.
 - The properties of the functions and sets we use are expressed by equations of the form $f \circ g = h$
- We assume that there are
 - Products (of sets; including the empty product),
 - Exponents (of sets),
 - The possibility of expressing subsets by characteristic functions
 - (Probably a bit more)

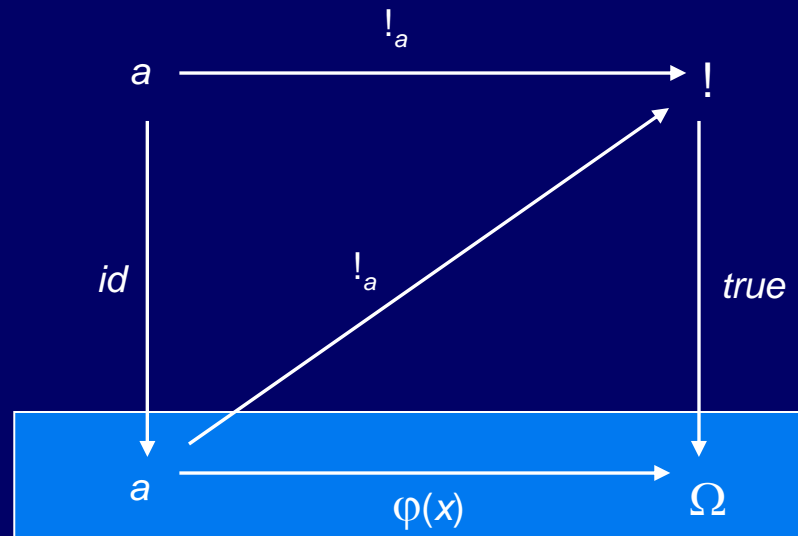


Logic and Topoi

- A structure defined by exactly these properties is (simplified) known as a *Topos*:
 - A category of *objects* and *arrows* with
 - Products
 - Coproducts
 - Exponents
 - A subobject classifier
- From the category theory point of view objects and arrows
 - are atomic
 - and are specified only by their behavior relative to other objects and arrows expressed by equations.
- To evaluate an expression all that is needed is the composition operation.
- Most of the necessary constructions can be found in (Goldblatt, 1979).

Logic and Topoi

- *Universal closure:*
if $\forall x: \varphi(x)$ is an axiom then the following diagram commutes



$$\forall \varphi(x) = true \Leftrightarrow \varphi(x) = !_a \circ true$$

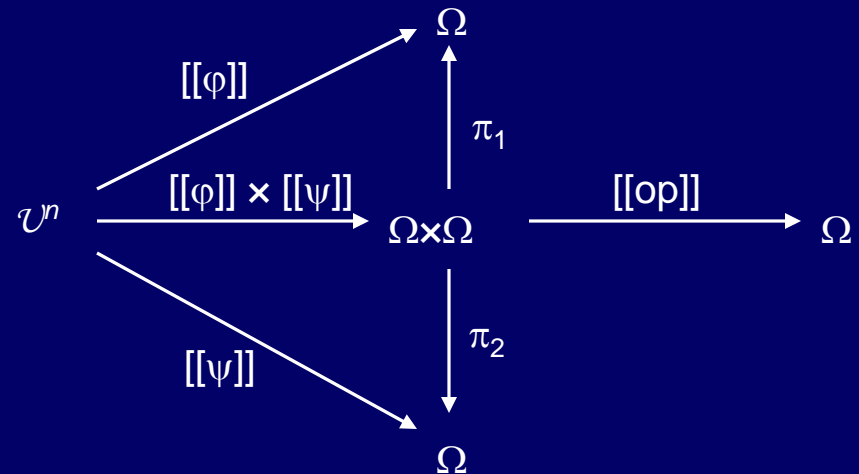
- This is a very restricted version of universal quantification, a general one is possible, but involves exponents and pullbacks

Logic and Topoi

- What do we need for an interpretation (cont.)
 - Logical connectives: functions from truth-values to truth-values

$$[[\wedge]], [[\vee]], [[\rightarrow]] : \Omega \times \Omega \rightarrow \Omega \qquad [[\neg]] : \Omega \rightarrow \Omega$$
 - Complex formulas:
 - $[[\neg\phi]] = [[\neg]] \circ [[\phi]]$
 - $[[\phi(x_1, \dots, x_n) \text{ op } \psi(x_1, \dots, x_n)]] = [[\text{op}]] \circ [[\phi]] \times [[\psi]]$
 - Here $[[\phi]] \times [[\psi]]$ refers to the product function:

- For logical operators it is possible to specify commuting diagrams determining their Topos interpretations.





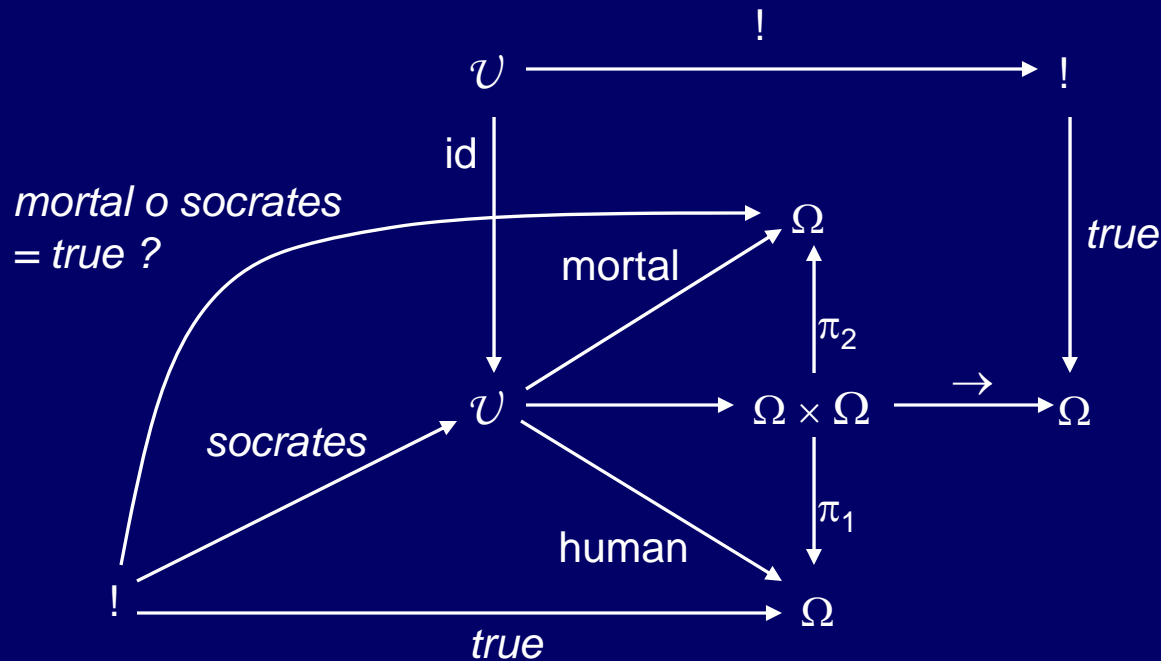
A Simple Example

- The theory consists of the following rules
 - Facts
 - *Socrates is human*
 - *The robot is not human*
 - Rules
 - *All humans are mortal*
 - *All mortal beings ascend to heaven*
 - *All beings in heaven are angels*
- We want to deduce that *Socrates is an angel*
- What about the *robot*? And what about an object *something* (where nothing is known about object *something*)?

Abstract Models

- The Socrates example:

- $\forall x : \text{human}(x) \rightarrow \text{mortal}(x) \Rightarrow \text{true} \circ (\text{human } x \text{ mortal}) = \text{true} \circ !$
- $\text{human}(\text{socrates}) \Rightarrow \text{human} \circ \text{socrates} = \text{true}$



+ all the other equations
in the diagram
(every triangle corresponds
to an equation)

+ universality of the product:
 $(\text{human} \circ \text{socrates})$
 $\times (\text{mortal} \circ \text{socrates}) =$
 $(\text{human } x \text{ mortal}) \circ \text{socrates}$

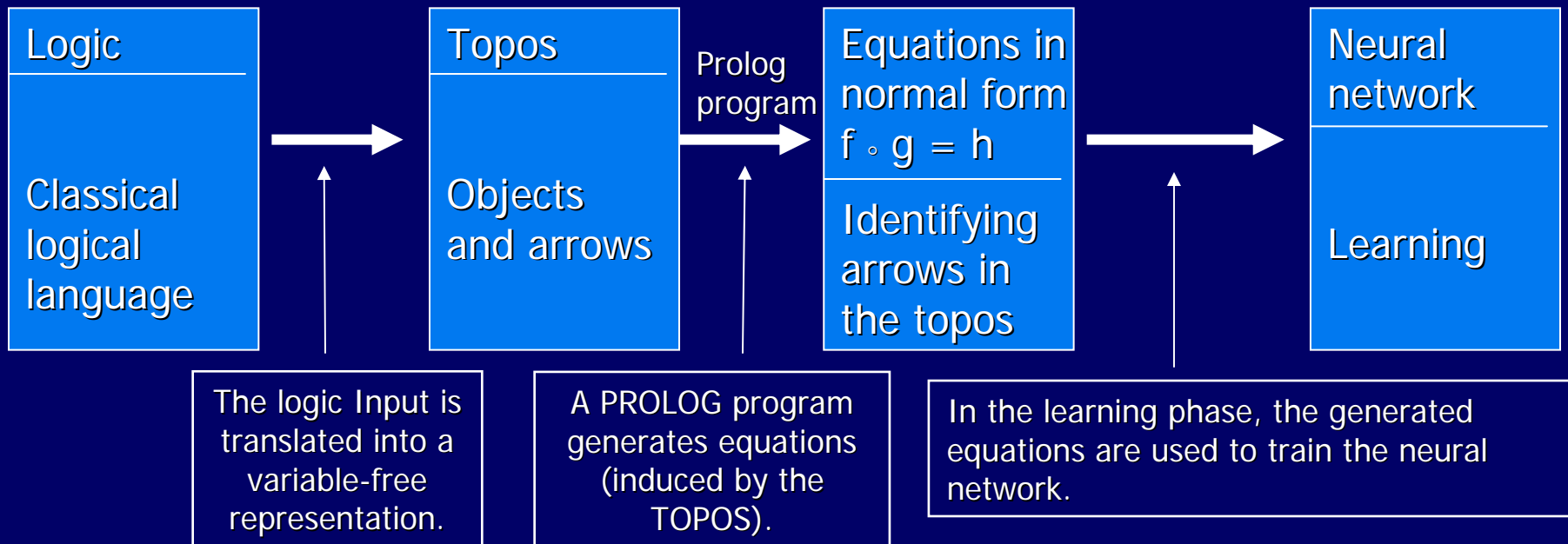


A Programming Language \mathcal{L}_T

\mathcal{L}_T	Intended Interpretation
$!$	terminal object $! = \mathcal{V}^0$
$\$\$$	truth-value object Ω
u	universe \mathcal{U}
$F :: X \dashrightarrow Y$	definition of an arrow F
$t, f :: ! \dashrightarrow \$\$$	truth-values <i>true</i> and <i>false</i>
$X \times Y$	product object/arrow of X and Y
$X \circ Y$	composition of arrows X and Y
$! :: _ \dashrightarrow !$	terminal arrows
$X :: X \dashrightarrow X$	Identities
$X = Y$	equation between arrows X and Y

Architecture

The account has the following general architecture





Neural Learning

- In order to enable the neural network to learn, some objects and arrows have (fixed) representations:
 - *true* ~ (1.0,0.0,0.0,.....)
 - *false* ~ (0.0,1.0,0.0,.....)
 - The truth values *true* and *false* are considered to be distinct.
 - All other objects and arrows are initialized with random values.

- The input of the network represents two arrows:
 - Domain of the first arrow
 - Representation of the first arrow
 - Codomain of the first arrow
 - Representation of the second arrow
 - Codomain of the second arrow

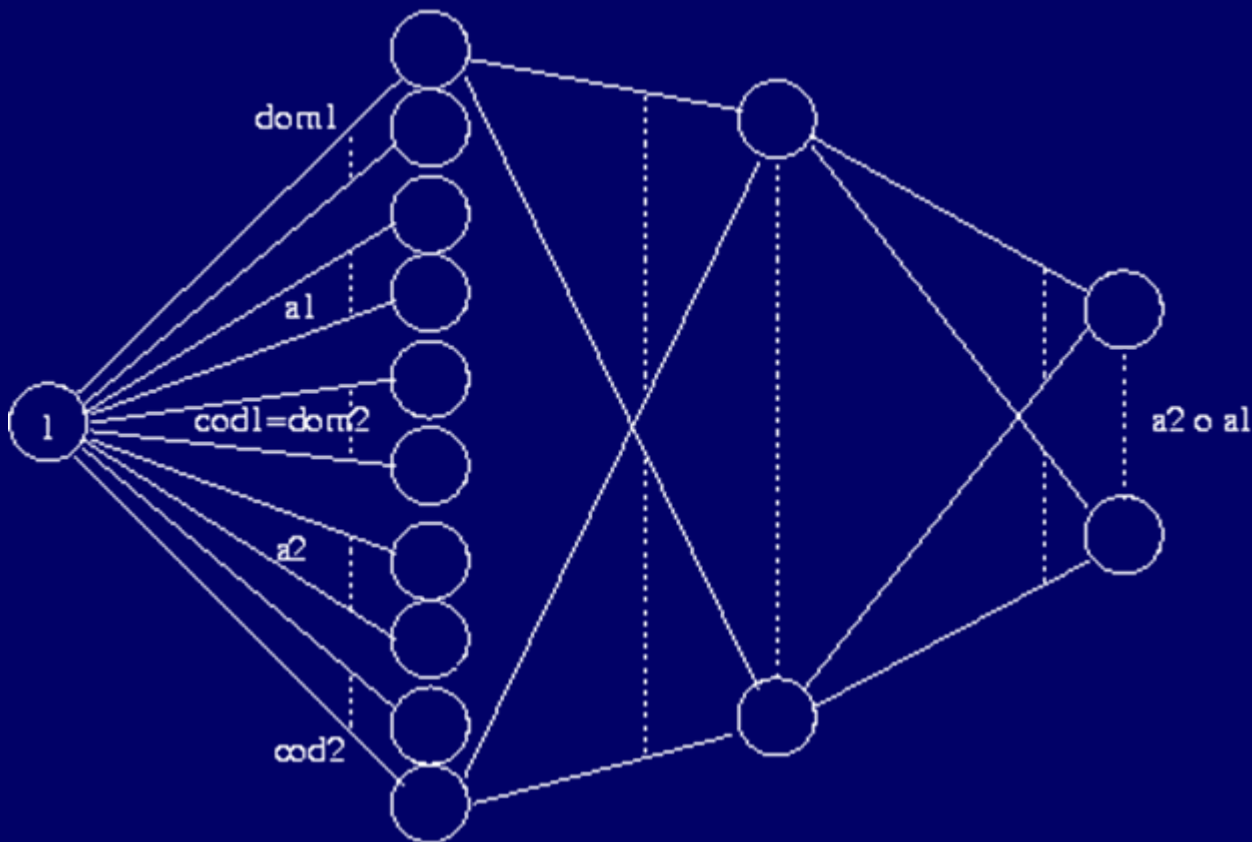
- The network is a feedforward network learning by backpropagation.

The Network

first layer: $5 \times n$

hidden layer: $? \times n$

output layer: n



Objects: points in $[0,1]^n$

Arrows: points in $[0,1]^n +$
domain and codomain

Computing: $a2 \circ a1 = b$

The input is represented
by weights from the
initial node with
activation 1



Example Code

□ “Background”

```
!.                               # the terminal object
$$                               # the truth value object
u.                               # the universe
static t:: ! ---> $$            # true
static f:: ! ---> $$            # false
not::      $$ ---> $$           # negation
->::      $$ x $$ ---> $$       # implication
not o t = f.
not o f = t.
-> o t x t = t.
-> o t x f = f.
-> o f x t = t.
-> o f x f = t.
```



Example Code

The Socrates example

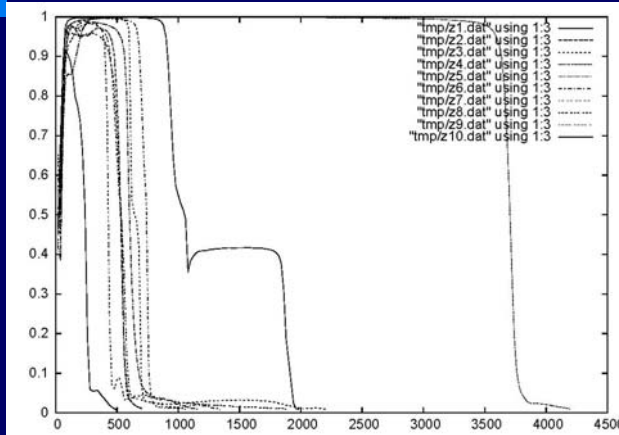
```
# predicates of the theory
human, mortal, heaven, angel:: u ---> $$ .
define X ==> Y:: -> o X x Y o d u = t o ! .
human ==> mortal .
mortal ==> heaven .
heaven ==> angel .
#individuals
distinctive
socrates, robot, something:: ! ---> u .
human o socrates = t .
human o robot = f .
```

(generates ~100 equations)

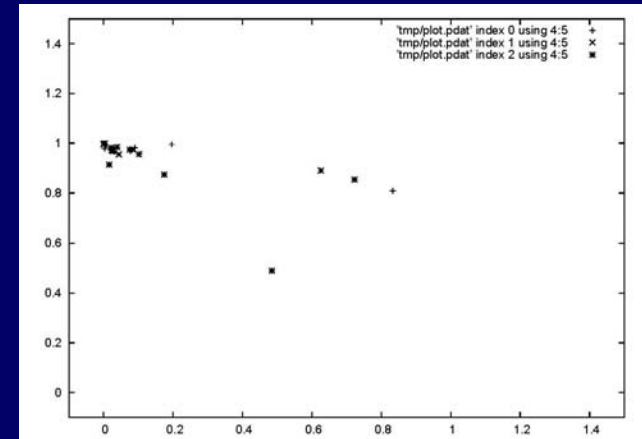
```
# test the learned theory
mortal o something = t ,
mortal o something = f ,
mortal o robot = t ,
mortal o robot = f ,
mortal o socrates = t ,
mortal o socrates = f ,
heaven o something = t ,
heaven o something = f ,
heaven o socrates = t ,
heaven o socrates = f ,
heaven o robot = t ,
heaven o robot = f ,
angel o something = t ,
angel o something = f ,
angel o socrates = t ,
angel o socrates = f ,
angel o robot = t ,
angel o robot = f .
```

The Socrates Example

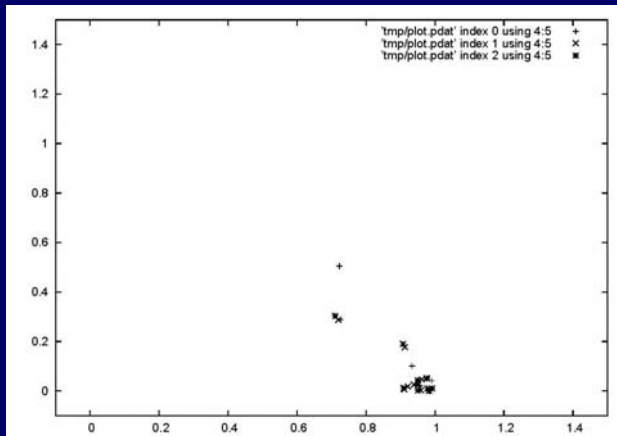
Maximal error.



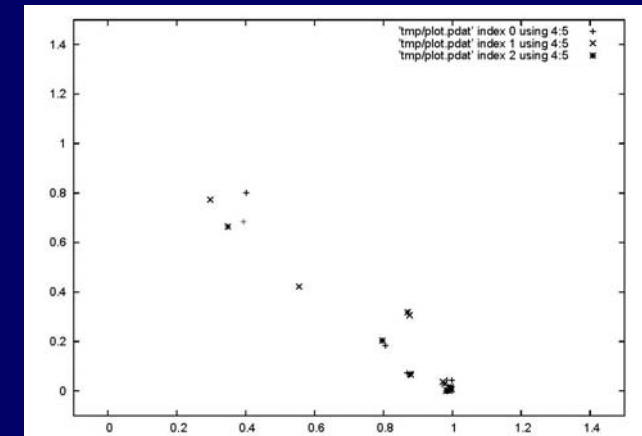
Robot is in heaven.



Socrates is in heaven.



Something is in heaven.





Conclusions

Consequences for Cognitive Science

- Solving inference task without an inference system
 - Cognitive psychology often claims that humans are no inference machines but reason in models
 - Connectionist learning of logic learns models of a theory but does not deduce anything
- Time critical decisions
 - A nice property of a trained neural network is the fact that it needs nearly no time to answer a query
 - Even for time critical applications logic-based theories can be used
- Uniformity
 - The network topology is fixed
 - Therefore, no construction of a new network is necessary if new input must be learned



Conclusions

- We sketched a framework for coding first-order logical inferences on the neural level.
- The architecture is robust because queries can be answered by the network even though no complete information may be available.
- Even in time-critical situations the network is able to answer.
- The architecture gives a first idea how an interaction between the symbolic and the neural level can be achieved.
- The architecture is cognitively more plausible than pure symbolic or sub-symbolic approaches.



Integrated Memory

- The topos level establishes a semi-symbolic level between the logical representation and the neural representation.
- Memory is coded implicitly by the topos constructions, and the weights of the neural network.
- Notice that not only an input is learned but an approximation of a logical model.
- The training is one-directional: from logical expressions, the neural representation is gained.
- The other direction, namely an extraction of rules from a trained network is currently not possible; nevertheless the network can be queried.



Thank you very much!!





References

□ Neuro-Symbolic Integration (Selection)

- Gust, H. & Kühnberger, K.-U. (2004). Cloning Composition and Logical Inference in Neural Networks Using Variable-Free Logic. *AAAI Fall Symposium Series 2004*, Symposium: Compositional Connectionism in Cognitive Science, Washington D.C., pp. 25-30.
- Gust, H. & Kühnberger, K.-U. (2006). Learning Symbolic Inferences with Neural Networks. In: B. Bara, L. Barsalou & M. Bucciarelli (Eds.): *CogSci 2005: XXVII Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, pp. 875-880.
- Gust, H., Kühnberger, K.-U. & Geibel, P. (2007). Learning Models of Predicate Logical Theories with Neural Networks Based on Topos Theory, in P. Hitzler & B. Hammer (eds.): *Perspectives of Neural-Symbolic Integration*, Series “Computational Intelligence”, Springer, pp. 209-240.
- Gust, H., Kühnberger, K.-U. & Geibel, P. (2008). Learning and Memorizing Models of Logical Theories in a Hybrid Learning Device, in M. Ishikawa, K. Doya, H. Miyamoto & T. Yamakawa (eds.): *Neural Information Processing*, 14th International Conference (ICONIP 2007), LNCS 4985, Springer, pp. 738-748.
- Gust, H., Kühnberger, K.-U. & Geibel, P. (2008). Perspectives of Neuro--Symbolic Integration – Extended Abstract –, *Recurrent Neural Networks 2008*.