

# Lecture 7: Instance-based Learning

## Cognitive Systems - Machine Learning

### Part II: Special Aspects of Concept Learning

*k*-nearest neighbors, locally weighted linear regression  
radial basis functions, lazy vs. eager learning  
Case-based Reasoning and Analogy

last change December 6, 2010

# Motivation

- all learning methods presented so far construct a general explicit description of the target function when examples are provided
- **Instance-based learning:**
  - ▶ examples are simply stored
  - ▶ generalization is postponed until a new instance must be classified
  - ▶ in order to assign a target function value, the example's relationship to the previously stored examples is examined
  - ▶ sometimes referred to as **lazy learning**

# Motivation

- **advantages:**

- ▶ instead of estimating  $f$  for the entire instance space, local approximations to the target function are possible
- ▶ especially if target function is complex but still decomposable

- **disadvantages:**

- ▶ classification costs are high
  - efficient techniques for indexing examples are important to reduce computational effort
- ▶ typically all attributes are considered when attempting to retrieve similar training examples
  - if the concept depends only on a few attributes, the truly most similar instances may be far away (“curse of dimensionality”)

# $k$ -nearest Neighbor Learning

- most basic instance-based method
- **assumption:**
  - ▶ instances correspond to a point in a  $n$ -dimensional space  $\mathcal{R}^n$
  - ▶ thus, nearest neighbors are defined in terms of the standard **Euclidean Distance**

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

where an instance  $x$  is described by  $\langle a_1(x), a_2(x), \dots, a_n(x) \rangle$

- target function may be either discrete-valued or real-valued

# k-nearest Neighbor Learning

- **discrete-valued target function:**

- ▶  $f : \mathfrak{R}^n \rightarrow V$  where  $V$  is the finite set  $\{v_1, v_2, \dots, v_s\}$
- ▶ the target function value is the most common value among the  $k$  nearest training examples

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^k \delta(v, f(x_i))$$

where

$$\delta(a, b) = 1 \text{ if } (a=b)$$

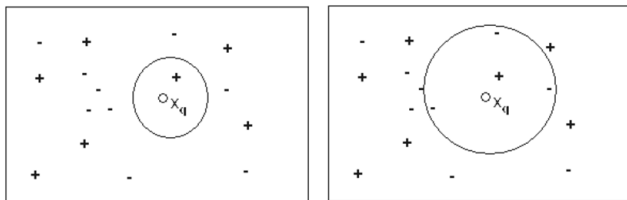
$$\delta(a, b) = 0 \text{ otherwise}$$

- **continuous-valued target function:**

- ▶ algorithm has to calculate the mean value instead of the most common value
- ▶  $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

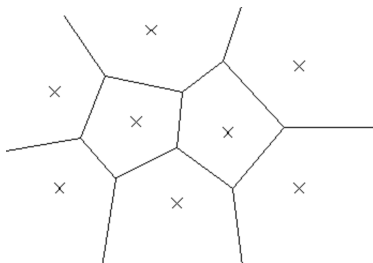
# k-nearest Neighbor Learning



- e.g. instances are points in a two-dimensional space where the target function is boolean-valued
  - ▶ 1-nearest neighbor:  $x_q$  is classified positive
  - ▶ 4-nearest neighbor:  $x_q$  is classified negative

# Hypothesis Space

- **no explicit** hypothesis is formed
- decision surface is a combination of convex polyhedra surrounding each of the training examples
- for each training example, the polyhedron indicates the set of possible query points  $x_q$  whose classification is completely determined by this training example (**Voronoi diagram**)



# Distance-Weighted Nearest Neighbor

- contribution of each of the  $k$  nearest neighbors is weighted according to their distance to  $x_q$ 
  - ▶ **discrete-valued target functions**

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

where  $w_i \equiv \frac{1}{d(x_q, x_i)^2}$  and  $\hat{f}(x_q) = f(x_i)$  if  $x_q = x_i$

- ▶ **continuous-valued target function:**

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$



# Locally Weighted Regression

- a note on terminology:
  - ▶ *Regression* means approximating a real-valued target function
  - ▶ *Residual* is the error  $\hat{f}(x) - f(x)$  in approximating the target function
  - ▶ *Kernel function* is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function  $K$  such that  $w_i = K(d(x_i, x_q))$
- nearest neighbor approaches can be thought of as approximating the target function at the single query point  $x_q$
- locally weighted regression is a generalization to this approach, because it constructs an explicit approximation of  $f$  over a local region surrounding  $x_q$

# Locally Weighted Linear Regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \dots + w_n a_n(x)$$

- methods like **gradient descent** can be used to calculate the coefficients  $w_0, w_1, \dots, w_n$  to minimize the error in fitting such linear functions
- ANNs require a global approximation to the target function
- here, just a local approximation is needed

⇒ the error function has to be redefined

# Locally Weighted Linear Regression

- possibilities to redefine the error criterion  $E$ 
  - Minimize the squared error over just the  $k$  nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2$$

- Minimize the squared error over the entire set  $D$ , while weighting the error of each training example by some decreasing function  $K$  of its distance from  $x_q$

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

- Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

# Locally Weighted Linear Regression

- choice of the error criterion
    - ▶  $E_2$  is the most esthetically criterion, because it allows every training example to have impact on the classification of  $x_q$
    - ▶ however, computational effort grows with the number of training examples
    - ▶  $E_3$  is a good approximation to  $E_2$  with constant effort
- Gradient descent rule

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest neighbors}} K(d(x_q, x))(f(x) - \hat{f}(x))a_j(x)$$

with  $\eta$  as learning rate,  $a_j$  as attribute value

# Locally Weighted Linear Regression

- Remarks on locally weighted linear regression:
  - ▶ in most cases, constant, linear or quadratic functions are used
  - ▶ costs for fitting more complex functions are prohibitively high
  - ▶ simple approximations are good enough over a sufficiently small subregion of  $X$

# Radial Basis Functions

- closely related to distance-weighted regression and to ANNs
- learned hypotheses have the form

$$\hat{f}(x) = w_0 + \sum_{u=1}^k w_u \cdot K_u(d(x_u, x))$$

where

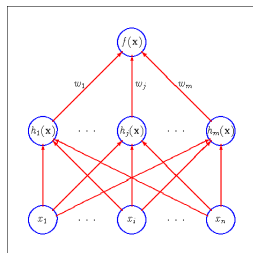
- ▶ each  $x_u$  is an instance from  $X$  and
  - ▶  $K_u(d(x_u, x))$  decreases as  $d(x_u, x)$  increases and
  - ▶  $k$  is a user-provided constant
- though  $\hat{f}(x)$  is a global approximation to  $f(x)$ , the contribution of each of the  $K_u$  terms is localized to a region nearby the point  $x_u$

# Radial Basis Functions

- it is common to choose each function  $K_u(d(x_u, x))$  to be a Gaussian function centered at  $x_u$  with some variance  $\sigma_u^2$

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2}d^2(x_u, x)}$$

- the function of  $\hat{f}(x)$  can be viewed as describing a two-layer network where the first layer of units computes the various  $K_u(d(x_u, x))$  values and the second layer a linear combination of the results



# Remarks

- highly effective inductive inference method for many practical problems provided a sufficiently large set of training examples
- robust to noisy data
- weighted average smoothes out the impact of isolated noisy training examples
- **inductive bias of  $k$ -nearest neighbors**
  - ▶ assumption that the classification of  $x_q$  will be similar to the classification of other instances that are nearby in the Euclidean Distance
- **curse of dimensionality**
  - ▶ distance is based on all attributes
  - ▶ in contrast to decision trees and inductive logic programming
  - ▶ solutions to this problem
    - attributes can be weighted differently
    - eliminate least relevant attributes from instance space



## Remarks on Lazy and Eager Learning

- **lazy methods** defer the decision of how to generalize beyond the training data until a new query instance  $x_q$  is encountered
- **eager methods** generalize before any new query instance is encountered
- Lazy methods allow stepwise changes of hypotheses by taking into account new examples

In contrast: in many eager methods hypotheses cannot be incrementally updated

- differences in computation time are obvious
  - essential differences in the **inductive bias**
    - ▶ lazy methods are able to consider the query instance  $x_q$  when deciding how to generalize
    - ▶ eager methods already have committed to a global approximation of the target function before any  $x_q$  is encountered
- ⇒ a lazy learner uses a richer  $H$ , because it uses many different local hypotheses to form a global approximation

## Lazy Learning – Related Approaches

- Lazy Learning is an approach for modeling **exemplar based categorization** (see human learning)
- Similarity based approach is related to cluster analytical approaches (***k*-means cluster analysis**)
- If *k*-means cluster are determined in the exemplar space, prototypes can be calculated as average cases (for metric features)
- For discrete-valued features (cp. family resemblance, human learning): contrast metric or Jaccard coefficient
- The *k*-nearest neighbor approach can also be used for **Case-based Reasoning** (CBR)

Jaccard Coefficient:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

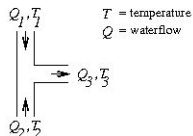
# Case-based Reasoning

- If standard inference methods cannot be applied (no formalized domain knowledge, highly complex search space), CBR and analogical reasoning are helpful
- CBR: Find the case (the  $k$  cases) most similar to the current case and use information from the retrieved case for the new case
- Example: For some car offered for sale, the price might be determined by averaging the prices of the cars with the most similar features (brand, model, milage, etc.)
- Example: CADET system (Sycara et al., 1992), design of water faucets
- CBR/Analogy can also be used for problem solving: Find the case with the most similar problem specification and transfer the solution

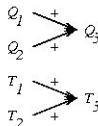
# Design of Water Faucets

**A stored case:** T-junction pipe

Structure:



Function:

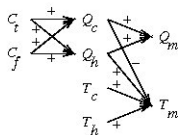


**A problem specification:** Water faucet

Structure:

?

Function:



# Ubiquitousness of Analogical Thinking

Analogy is a powerful and often used cognitive skill:

Exploiting experience from one domain (**base**)  
to explain/predict unknown aspects of or solve problems  
in a different domain (**target**).

*Analogy pervades all our thinking, our everyday speech and our trivial conclusions as well as artistic ways of expression and the highest scientific achievements.*

*Polya, How to Solve It, 1945*

# Analogy as a Central Research Topic in Cognitive Science

e.g., Gentner, D., Holyoak, K.J., and Kokinov, B. (2001). *The Analogical Mind – Perspectives from Cognitive Science*. MIT Press.

- Cognitive Simulation Models (Psychology, AI)
- Emotional Analogies, Scientific Discovery (Philosophy, Psychology, AI)
- Metaphors and Analogy (Linguistics, Philosophy, Mathematics)
- Neurocognitive Basis of Relational Reasoning (Neuropsychology)
- Analogy in Education (Psychology)
- Analogy in (Software) Engineering (HCI, AI)
- ...

## Example 1: Analogy in Literature

*Analogy or proportion is when the second term is to the first as the fourth to the third. We may then use the fourth for the second, or the second for the fourth. Sometimes too we qualify the metaphor by adding the term to which the proper word is relative. [...] As old age is to life, so is evening to day. Evening may therefore be called, 'the old age of the day,' and old age, 'the evening of life,' or, in the phrase of Empedocles, 'life's setting sun.'*

*Aristotle, Poetics, chap. 21, "Words"*

## Example 2: Analogy in History and Politics

End of war in Iraq – End of World War II

Global investor – locust

*Die Methode, nach der Analogie zu schließen, ist, wie überall, so auch in der Geschichte ein mächtiges Hilfsmittel; aber sie muß durch einen erheblichen Zweck gerechtfertigt, und mit ebensoviel Vorsicht als Beurteilung in Ausübung gebracht werden.*

*Was heisst und zu welchem Ende studiert man Universalgeschichte?  
Akademische Antrittsrede von Friedrich Schiller  
am 26.5.1789 in Jena*

*Used to transport negative/positive **emotions!***



## Example 2: Analogy in History and Politics

End of war in Iraq – End of World War II

Global investor – locust

*Die Methode, nach der Analogie zu schließen, ist, wie überall, so auch in der Geschichte ein mächtiges Hilfsmittel; aber sie muß durch einen erheblichen Zweck gerechtfertigt, und mit ebensoviel Vorsicht als Beurteilung in Ausübung gebracht werden.*

*Was heisst und zu welchem Ende studiert man Universalgeschichte?  
Akademische Antrittsrede von Friedrich Schiller  
am 26.5.1789 in Jena*

*Used to transport negative/positive emotions!*

## Example 2: Analogy in History and Politics

End of war in Iraq – End of World War II

Global investor – locust

*Die Methode, nach der Analogie zu schließen, ist, wie überall, so auch in der Geschichte ein mächtiges Hilfsmittel; aber sie muß durch einen erheblichen Zweck gerechtfertigt, und mit ebensoviel Vorsicht als Beurteilung in Ausübung gebracht werden.*

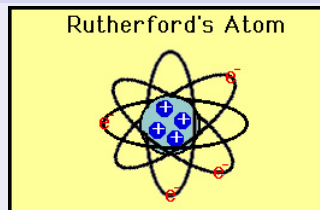
*Was heisst und zu welchem Ende studiert man Universalgeschichte?  
Akademische Antrittsrede von Friedrich Schiller  
am 26.5.1789 in Jena*

*Used to transport negative/positive **emotions!***

## Example 3: Scientific Analogy

The structure of the hydrogen atom  
is like  
the structure of the solar system.

Rutherford, 1911



*Analogy as source of scientific and artistic **creativity**.*

# Analogy in Education

Prove: the product of two even numbers is an even number

The product of two odd numbers is an odd number.

Proof:

- Odd number:  $even + 1 = 2N + 1$

- Product:

$$(2N + 1) \cdot (2M + 1) = 4NM + 2N + 2M + 1 = 2(2NM + N + M) + 1$$

## Analogy

- Even number:  $2N$

- Product:  $2N \cdot 2M = 4NM = 2(2NM)$

# Analogy in Education

Prove: the product of two even numbers is an even number

The product of two odd numbers is an odd number.

Proof:

- Odd number:  $even + 1 = 2N + 1$

- Product:

$$(2N + 1) \cdot (2M + 1) = 4NM + 2N + 2M + 1 = 2(2NM + N + M) + 1$$

## Analogy

- Even number:  $2N$

- Product:  $2N \cdot 2M = 4NM = 2(2NM)$

# Analogy in Education

Prove: the product of two even numbers is an even number

The product of two odd numbers is an odd number.

Proof:

- Odd number:  $even + 1 = 2N + 1$

- Product:

$$(2N + 1) \cdot (2M + 1) = 4NM + 2N + 2M + 1 = 2(2NM + N + M) + 1$$

## Analogy

- Even number:  $2N$

- Product:  $2N \cdot 2M = 4NM = 2(2NM)$

# Analogy in Software Engineering

## Code reuse vs developing from scratch

Systematic support of reuse can reduce development costs dramatically

Defense Information Systems Agency, 1995

In short, the business case for reuse consists of avoiding 80% of the development costs for reused components (plus some additional maintenance savings) minus the 50% extra it costs to build the reusable component in the first place.

Jeffrey Poulin and Brent Carlson, Computerworld, Feb 2004

# Analogy and Similarity

	Attributes mapped to target	Relations mapped to target	Example
Mere Appearance	Many	Few	A sunflower is like the sun
Literal Similarity	Many	Many	The K5 solar system is like our solar system
Analogy	Few	Many	The atom is like our solar system
Abstraction	Few	Many	The atom is a central force system
Metaphor	x	x	<i>She is the sun of my life</i>

(Gentner, 1983, 1997)



# Three Kinds of Analogy

## Proportional

- A is to B as C is to ?D
- **Evening is to Day as Old Age is to Life**  $\leftrightarrow$  "last part of"
- most simple form of analogy: transfer of *one* relation

## Predictive/Explanatory

- Carry-over of known principles to a new domain of interest
- **Rutherford Analogy**

## in Problem Solving

- *within domain*, use of examples (cf. CBR)
- Transfer of a known solution to a new problem
- **Mathematical/programming problems**

# Three Kinds of Analogy

## Proportional

- A is to B as C is to ?D
- **Evening is to Day as Old Age is to Life**  $\leftrightarrow$  "last part of"
- most simple form of analogy: transfer of *one* relation

## Predictive/Explanatory

- Carry-over of known principles to a new domain of interest
- **Rutherford Analogy**

## in Problem Solving

- *within domain*, use of examples (cf. CBR)
- Transfer of a known solution to a new problem
- **Mathematical/programming problems**

# Three Kinds of Analogy

## Proportional

- A is to B as C is to ?D
- **Evening is to Day as Old Age is to Life**  $\leftrightarrow$  "last part of"
- most simple form of analogy: transfer of *one* relation

## Predictive/Explanatory

- Carry-over of known principles to a new domain of interest
- **Rutherford Analogy**

## in Problem Solving

- *within domain*, use of examples (cf. CBR)
- Transfer of a known solution to a new problem
- **Mathematical/programming problems**

# Subprocesses in Analogical Reasoning

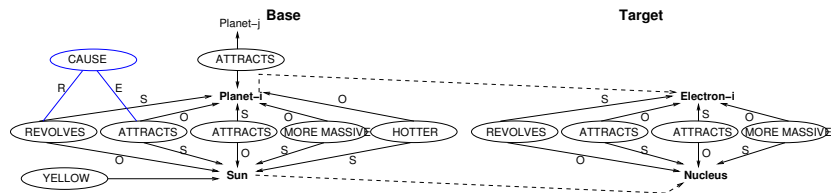
## Overview

- Representation
- Retrieval
- Mapping
- Inference/Transfer
- Learning

(in all cognitive theories: Gentner, Holyoak, Keane, ...)

# Representation

- Structural representation (graph, term, semantic net, ...)
- Problem: Representation crucial for mapping success
- $on(a,b)$  vs.  $below(b,a)$ ;  $x > y$  vs.  $x - y > 0$

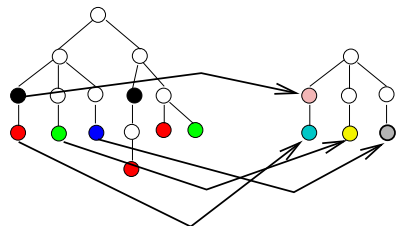


# Retrieval

- Bottleneck of analogy
- Governed by superficial similarity (feature based measures)
- Novices fail to identify useful base problem (experiments by Novick, 1988)
- In Education: Present suitable base problems explicitly
- In Computation: Use structural similarity for retrieval

# Mapping

- Core of analogy
- Structure preserving mapping
- First order (same relations/functions in both domains) or higher order
- In well-structured, formal domains: homomorphism



# Homomorphism

- Structure preserving mapping  $f : S \rightarrow T$
- such that  $f(o_S(s_1, \dots, s_n)) = o_T(f(s_1), \dots, f(s_n))$

$$\begin{array}{ccc} S_1 \times \dots \times S_n & \xrightarrow{o_S} & S \\ \downarrow f & = & \downarrow f \\ T_1 \times \dots \times T_n & \xrightarrow{o_T} & T \end{array}$$



# Homomorphism

- Structure preserving mapping  $f : S \rightarrow T$
- such that  $f(o_S(s_1, \dots, s_n)) = o_T(f(s_1), \dots, f(s_n))$

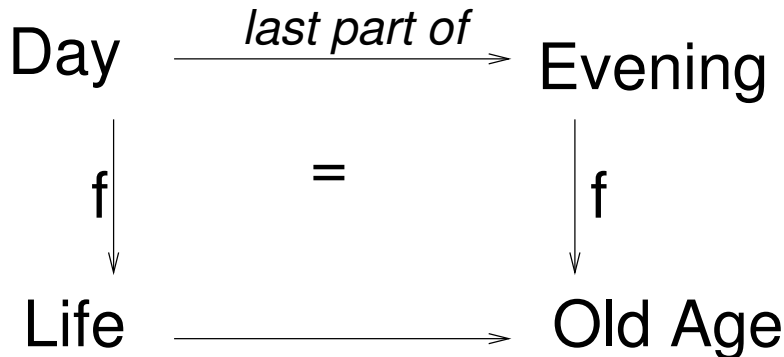
$$\begin{array}{ccc} S_1 \times \dots \times S_2 & \xrightarrow{o_S} & S \\ \downarrow f & = & \downarrow f \\ T_1 \times \dots \times T_2 & \xrightarrow{o_T} & T \end{array}$$

# Homomorphism

- Structure preserving mapping  $f : S \rightarrow T$
- such that  $f(o_S(s_1, \dots, s_n)) = o_T(f(s_1), \dots, f(s_n))$

$$\begin{array}{ccc} S_1 \times \dots \times S_2 & \xrightarrow{o_S} & S \\ \downarrow f & = & \downarrow f \\ T_1 \times \dots \times T_2 & \xrightarrow{o_T} & T \end{array}$$

## Example 1: Proportional Analogy



## Example 2: Proportional Analogy

Relation between string concatenation and length of strings

$$\begin{array}{ccc} W_1 \times W_2 & \xrightarrow{\circ} & W \\ \downarrow \mathbf{f} & = & \downarrow \mathbf{f} \\ N_1 \times N_2 & \xrightarrow{+} & N \end{array}$$

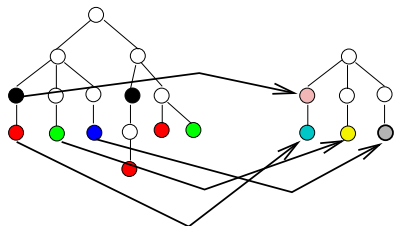
## Example 2: Proportional Analogy

Relation between string concatenation and length of strings

$$\begin{array}{ccc} \text{the x me} & \xrightarrow{\circ} & \text{theme} \\ \downarrow \mathbf{f} & = & \downarrow \mathbf{f} \\ 3 \text{ x } 2 & \xrightarrow{+} & 5 \end{array}$$

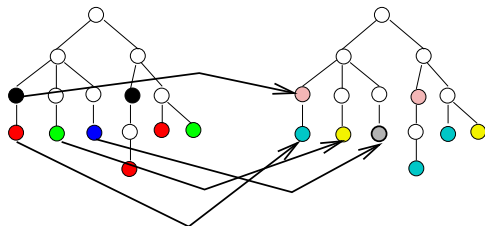
# Transfer

- Based on mapping
- Carry-over of information from base to target
- Inference of unknown characteristics of target
- Transfer/adaptation of a solution



# Transfer

- Based on mapping
- Carry-over of information from base to target
- Inference of unknown characteristics of target
- Transfer/adaptation of a solution







# Learning

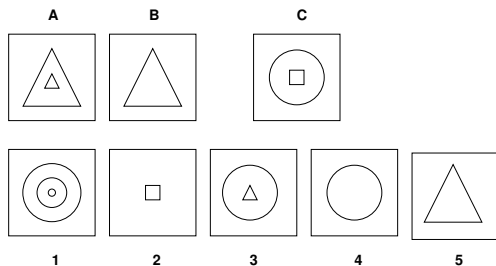
- Acquisition of more general schemes or rules by abstraction (Solar system, atom  $\leftrightarrow$  central force system)
- Analogy as “beginner’s strategy”: Acquisition of general concepts makes analogical reasoning obsolete!

# Computational Cognitive Science Systems

- Forbus, Falkenhainer & Gentner (1989): SME  
Naive physics
- Hummel & Holyoak (1997): LISA  
Problem solving (between domain)
- Anderson & Thompson (1989): modified ACT  
Programming/geometrical proofs (within domain)

# Cognitive AI Systems

- Evans (1968): Geometrical analogies (intelligence test)
- Veloso & Carbonell (1993): Plan construction (Prodigy)
- O'Hara (1992): geometrical analogies (PAN)
- Hofstadter (1995): letter strings (Copycat)



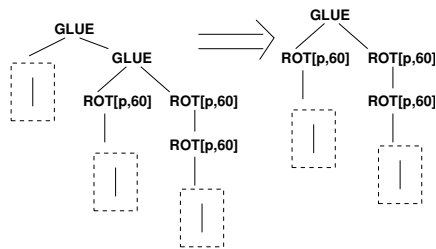
# Cognitive AI Systems

- Evans (1968): Geometrical analogies (intelligence test)
- Veloso & Carbonell (1993): Plan construction (Prodigy)
- O'Hara (1992): geometrical analogies (PAN)
- Hofstadter (1995): letter strings (Copycat)



# Cognitive AI Systems

- Evans (1968): Geometrical analogies (intelligence test)
- Veloso & Carbonell (1993): Plan construction (Prodigy)
- O'Hara (1992): geometrical analogies (PAN)
- Hofstadter (1995): letter strings (Copycat)



- Evans (1968): Geometrical analogies (intelligence test)
- Veloso & Carbonell (1993): Plan construction (Prodigy)
- O'Hara (1992): geometrical analogies (PAN)
- Hofstadter (1995): letter strings (Copycat)

abc : abd :: kji : ??

# Cognitive Plausibility of Approaches

- Unflexible representation  
(additional re-representation mechanisms in PAN and Copycat)
- Mapping of graphs  $\hat{=}$  subgraph isomorphism problem (NP-hard)  
(heuristics in SME and LISA)
- Transfer only carry-over and replace,  
no real adaptation (permutation, deletion)
- Learning not addressed  
or modelled by an additional mechanism (ACT)

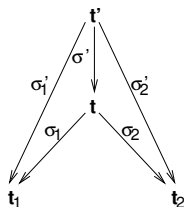
# Our Approach

- Anti-Unification (Reynolds, 1970): Mapping via common abstraction
- Mapping is governed directly by the common structure of base and target
- Allows use of equational theories for re-representation (Schmid, Gust, Kühnberger, Burghardt, 2003)
- Defined for first- and second-order case (Hasker, 1995)



# AU Example

- $t_1 = \text{attracts}(\text{sun}, \text{planet-}i)$
- $t_2 = \text{attracts}(\text{nucleus}, \text{electron-}i)$
- can be generalized to  
 $t = \text{attracts}(\text{central-body}, \text{orbiter})$   
where 'central-body' and 'orbiter' are variables
- Calculating the abstraction results in the mapping  
 $\varphi = \{(\text{sun}, \text{nucleus}) \mapsto \text{central-body},$   
 $(\text{planet-}i, \text{electron-}i) \mapsto \text{orbiter}\}$
- Thereby, the necessary substitutions of variables by constants are also known:  
 $\sigma_1 = \{ \text{central-body} \mapsto \text{sun}, \text{orbiter} \mapsto \text{planet-}i \}$  and  
 $\sigma_2 = \{ \text{central-body} \mapsto \text{nucleus}, \text{orbiter} \mapsto \text{electron-}i \}$
- Now the mapping of *sun* onto *nucleus* can be performed by applying first  $\varphi$  and then  $\sigma_2$ .



# First-Order AU Modulo Equational-Theories

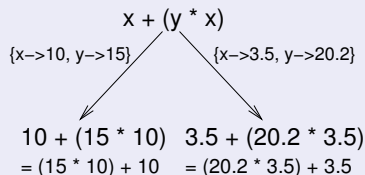
- Syntactic AU of  $10 + (15 \cdot 10)$  and  $(20.2 \cdot 3.5) + 3.5$  is  $x + y$
- The fact, that each term is the addition of a constant and a product containing this constant got lost in generalization!
- Equational theories can be used to model knowledge about re-representations in a natural way

## Simple Theory

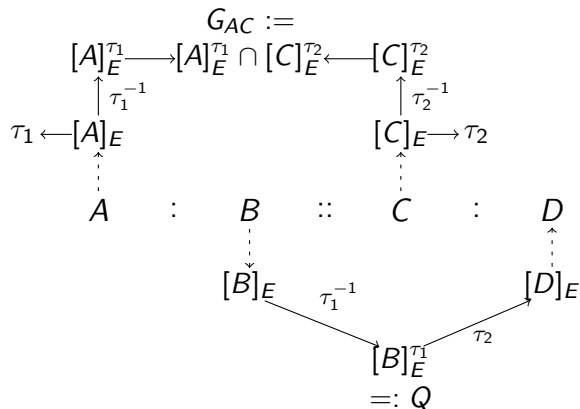
$$x + 0 =_E x$$

$$x + y =_E y + x$$

$$x + (y + z) =_E (x + y) + z$$



# Solving Proportional Analogies with E-Generalization



(BA Stephan Weller)

## 2nd Order Anti-Unification

- 1st order AU of  $10 + (15 \cdot 10)$  and  $3.5 - (20.2 \cdot 3.5)$  is  $x$
- The fact, that each term is built by an operation over a constant and a product got lost in generalization!
- 2nd order AU can be used to model generalization over function/predicate symbols

- Result:  $x F (y \cdot x)$   
where  $x$  and  $y$  are object variables and  $F$  is a function variable
- Applied to programming by analogy (Wagner and Schmid) in context of our work on inductive program synthesis techniques

# Programming by Analogy

**Fac-Problem:** If the factorial of 3 is calculated as  $3 \cdot 2 \cdot 1 \cdot 1$   
what is the factorial for a natural number  $n$ ?

**Fac-Solution:**  $fac(n) = if(n=0,1,n \cdot fac(n-1))$ .

**NSum-Problem:** If the neg. sum of 3 is calculated as  $((0 - 1) - 2) - 3$  what is  
the neg. sum for a natural number  $n$ ?

# Programming by Analogy

**Fac-Problem:** If the factorial of 3 is calculated as  $3 \cdot 2 \cdot 1 \cdot 1$   
what is the factorial for a natural number  $n$ ?

**Fac-Solution:**  $fac(n) = if(n=0,1,n \cdot fac(n-1))$ .

**NSum-Problem:** If the neg. sum of 3 is calculated as  $((0 - 1) - 2) - 3$  what is  
the neg. sum for a natural number  $n$ ?

## **Fac-Unfolding:**

if  $n = 0$  then 1  
else if  $n = 1$  then  $1 \cdot 1$   
else if  $n = 2$  then  $2 \cdot (1 \cdot 1)$   
else if  $n = 3$  then  $3 \cdot (2 \cdot (1 \cdot 1))$

## **NSum-Problem:**

if  $n = 0$  then 0  
else if  $n = 1$  then  $0 - 1$   
else if  $n = 2$  then  $(0 - 1) - 2$   
else if  $n = 3$  then  $((0 - 1) - 2) - 3$

# Programming by Analogy

## Fac-Unfolding:

```
if    n = 0 then 1
else if n = 1 then 1 · 1
else if n = 2 then 2 · (1 · 1)
else if n = 3 then 3 · (2 · (1 · 1))
```

## Fac-NSum-Generalization:

```
if    n=0 then x
else if n=1 then 1 F x
else if n=2 then 2 F (1 F x)
else if n=3 then 3 F (2 F (1 F x))
```

## NSum-Problem:

```
if    n = 0 then 0
else if n = 1 then 0 - 1
else if n = 2 then (0 - 1) - 2
else if n = 3 then ((0 - 1) - 2) - 3
```

$$\sigma_{\text{fac}} = \{x \mapsto 1, F \mapsto (\cdot \pi_1, \pi_2)\}$$

$$\sigma_{\text{nsum}} = \{x \mapsto 0, F \mapsto (- \pi_2, \pi_1)\}$$

*Note that arguments of subtraction  
op are reversed!*

# Programming by Analogy

## Fac-NSum-Generalization:

if  $n=0$  then  $x$

else if  $n=1$  then  $1 F x$

else if  $n=2$  then  $2 F (1 F x)$

else if  $n=3$  then  $3 F (2 F (1 F x))$

$$\sigma_{\text{fac}} = \{x \mapsto 1, F \mapsto (\cdot \pi_1, \pi_2)\}$$

$$\sigma_{\text{nsum}} = \{x \mapsto 0, F \mapsto (- \pi_2, \pi_1)\}$$

*Note that arguments of subtraction  
op are reversed!*

- The abstract term captures the role of 1 and 0 as neutral element and of  $\cdot$  and  $-$  as ‘combination-operator’ respectively.
- Obtaining the target solution: by applying the found substitutions to the recursive solution of the base problem
- Fac-Solution:  $\text{fac}(n) = \text{if}(n=0, 1, n \cdot \text{fac}(n-1))$ .
- NSum-Solution:  $\text{nsum}(n) = \text{if}(n=0, 0, \text{nsum}(n-1) - n)$ .



# Evaluation

- AU is applicable to model all kinds of analogy
- By including equational theories, background knowledge can be included in a natural way
- Mapping via abstraction vs. direct mapping
  - ▶ Computational advantage  
(getting rid of subgraph isomorphism problem)
  - ▶ Cognitive plausibility  
(mapping guided by the common role of objects)
- Re-representation is modeled by taking into account all possible representations wrt a given theory simultaneously
- Learning by abstraction is a side-effect of analogical reasoning

# Summary

- instance-based learning simply stores examples and postpones generalization until a new instance is encountered
- able to learn discrete- and continuous-valued concepts
- noise in the data is allowed (smoothed out by weighting distances)
- **Inductive Bias of  $k$ -nearest neighbors**: classification of an instance is similar to the classification of other instances nearby in the Euclidean Distance
- Locally Weighted Regression forms a local approximation of the target function
- Case-based reasoning (CBR) is a related approach to lazy learning
- Analogy is a process which can be used in CBR, analogy can incorporate (eager) generalization learning

# Learning Terminology

## Lazy Learning

<b>Supervised Learning</b>	<b>unsupervised learning</b>
----------------------------	------------------------------

Approaches:

<b>Concept / Classification</b>	Policy Learning
<b>symbolic</b>	<b>statistical</b> / neuronal network
<b>inductive</b>	analytical

Learning Strategy:

⇒ **learning from examples**