

Knowledge Extraction from Trained Neural Networks

- **An integral part of any Neural-Symbolic Learning System**
- **An important research area in its own right: explanation, integration with other systems, incremental learning**

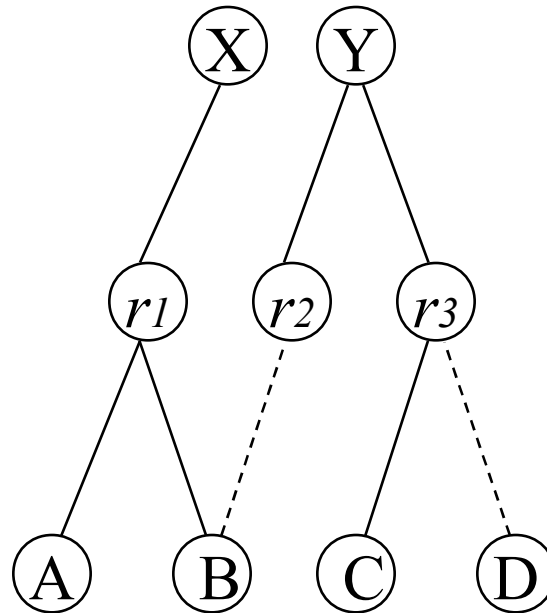
The Extraction Problem

Logic Program P:

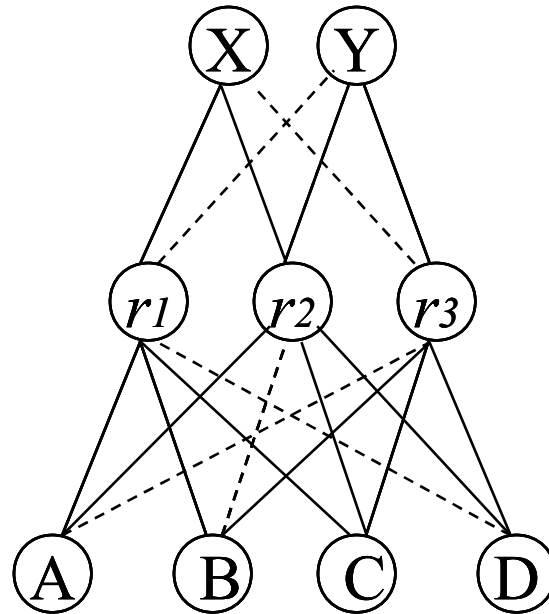
X if A,B;

Y if ~B;

Y if C,~D



The Extraction Problem



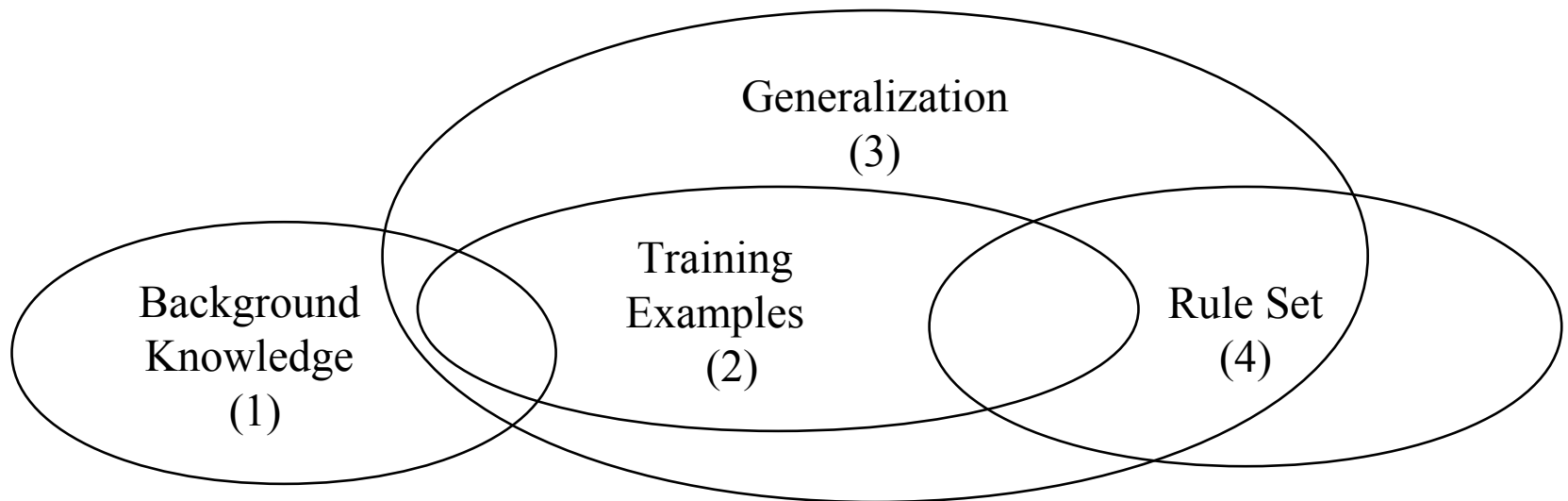
Logic Program P':



?

Quality x Complexity Trade-off

- How well the extracted rules reflect the network vs. computational complexity of extraction: time complexity (exponential) and space (size of rule set)

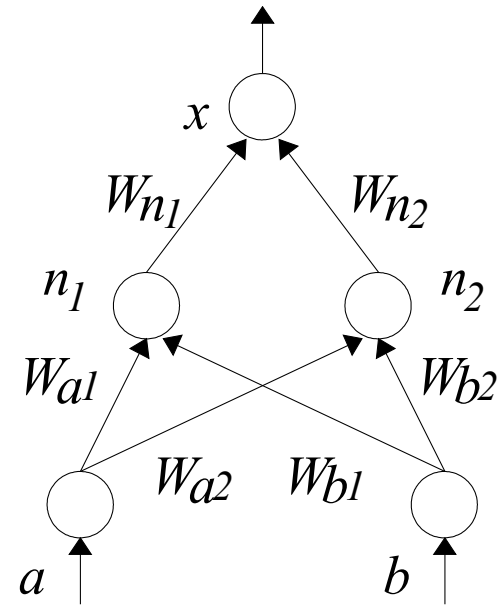


Extraction Properties

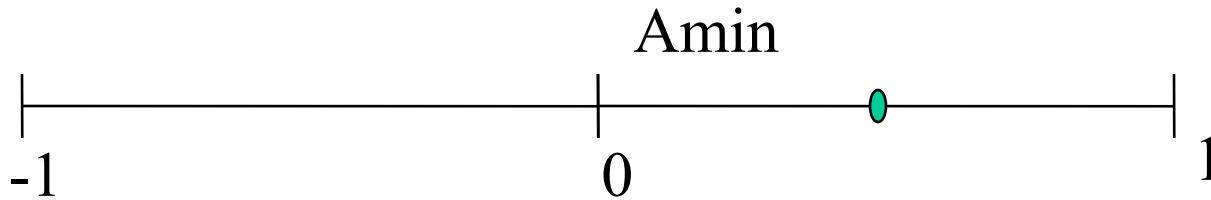
- Soundness: Each extracted rule is encoded in the network
- Completeness: All rules encoded in the network can be extracted
- Readability: As few rules as possible with as few antecedents as possible are extracted

Decompositional x Pedagogical ?

- Decompositional: split network into subnetworks and extract rules for each hidden and output neuron
- Pedagogical: treat network as black box and extract rules directly from input to output by querying the network
- Decompositional methods can be unsound and incomplete
- Sound and complete pedagogical methods have exponential worst-case complexity



The Unsoundness of Decompositional Approaches



- Towell and Shavlik's MofN approximates weights;
- Setiono's approach is based on pruning the network's weights;
- Fu's approach uses $\{0,1\}$ as activation of hidden neurons;
- Thrun's approach is sound but does not treat nonmonotonicity.

CILP Extraction Approach

To reduce the Quality x Complexity trade-off: *A Partial Ordering* on the Set of Input Vectors

Reduce Complexity by using some pruning rules

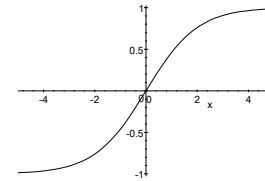
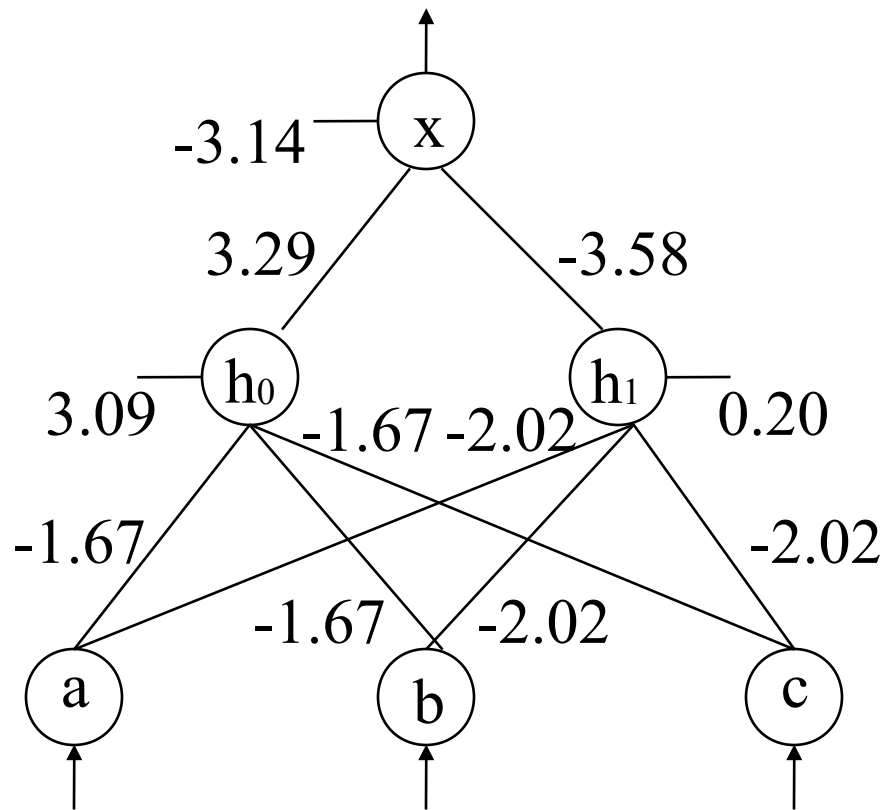
Enhance Readability by rule simplification

Maintain Quality by proving soundness

In the case of large networks, forego completeness for efficiency, but keep quasi-completeness

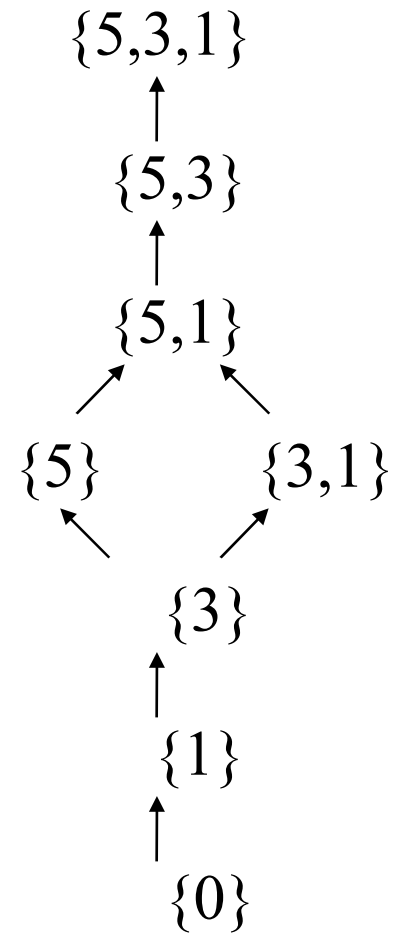
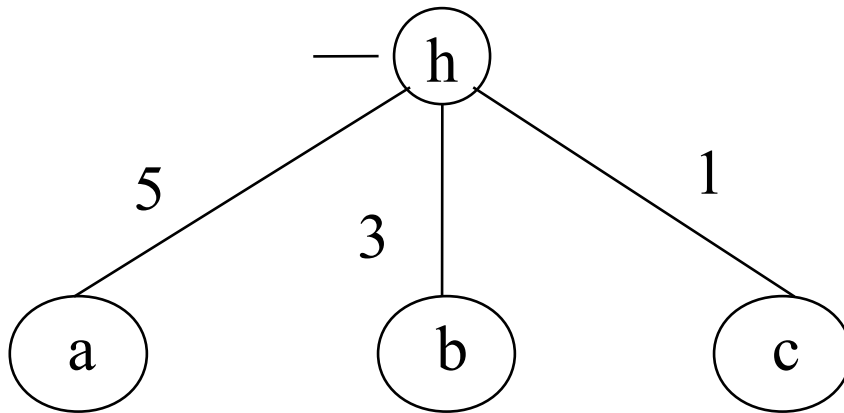
Extraction - An Example

- Exactly 2 out of $\{a,b,c\} \rightarrow x$



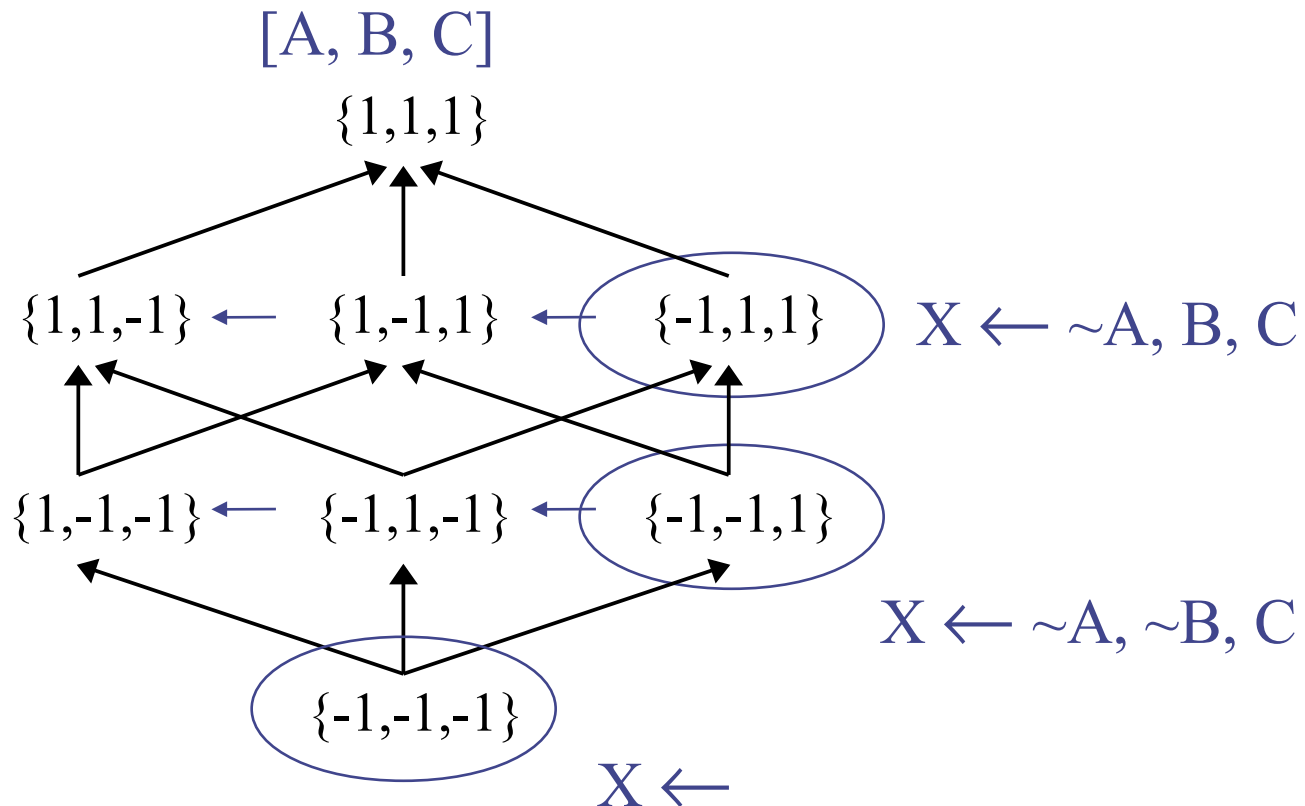
Sometimes it is easy to find the partial ordering

- By inspecting the weights, e.g. $W_a > W_b > W_c$

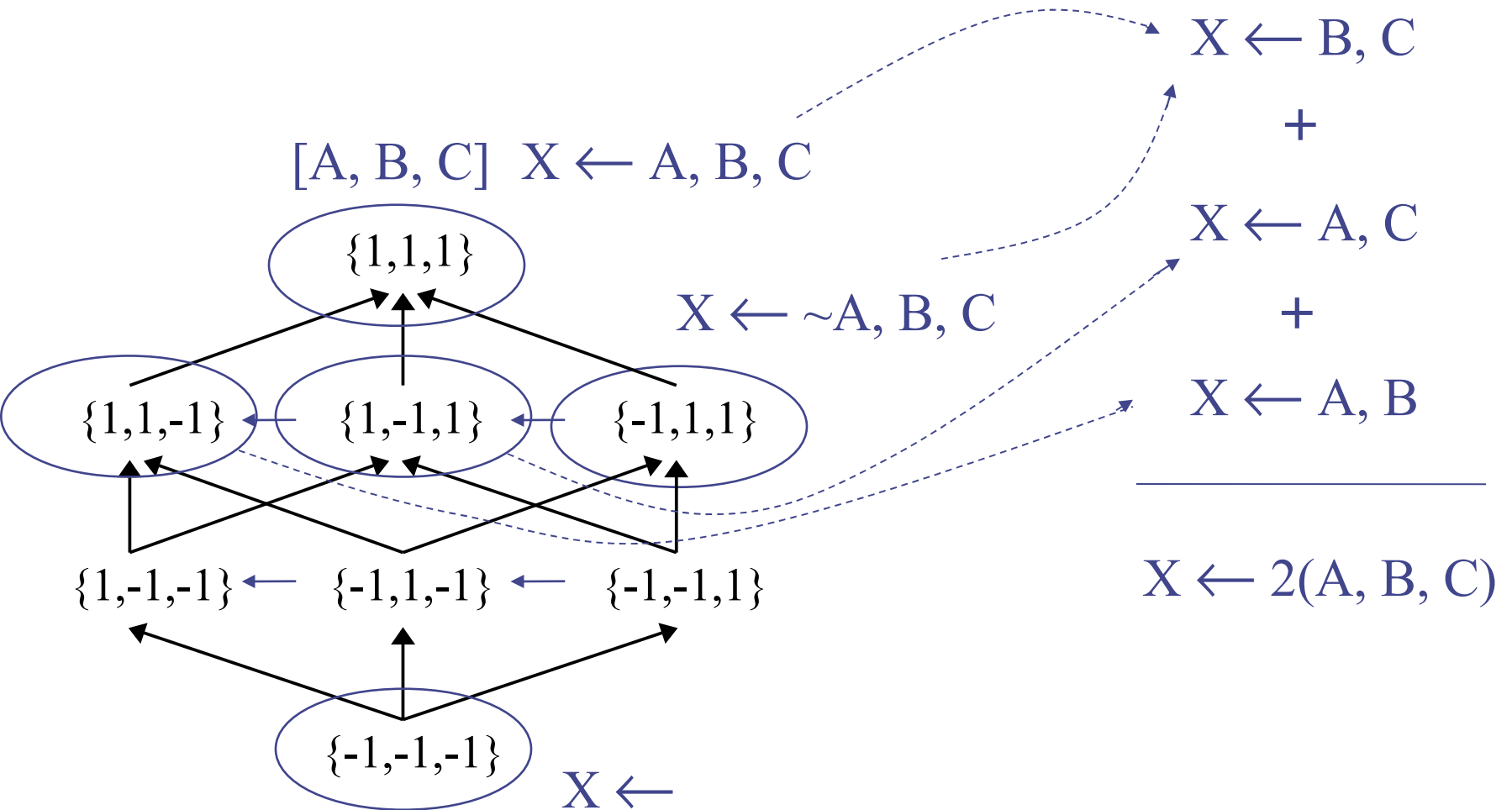


Partial Order on Input Vectors

- Consider a network with three inputs $\{A, B, C\}$ and output X

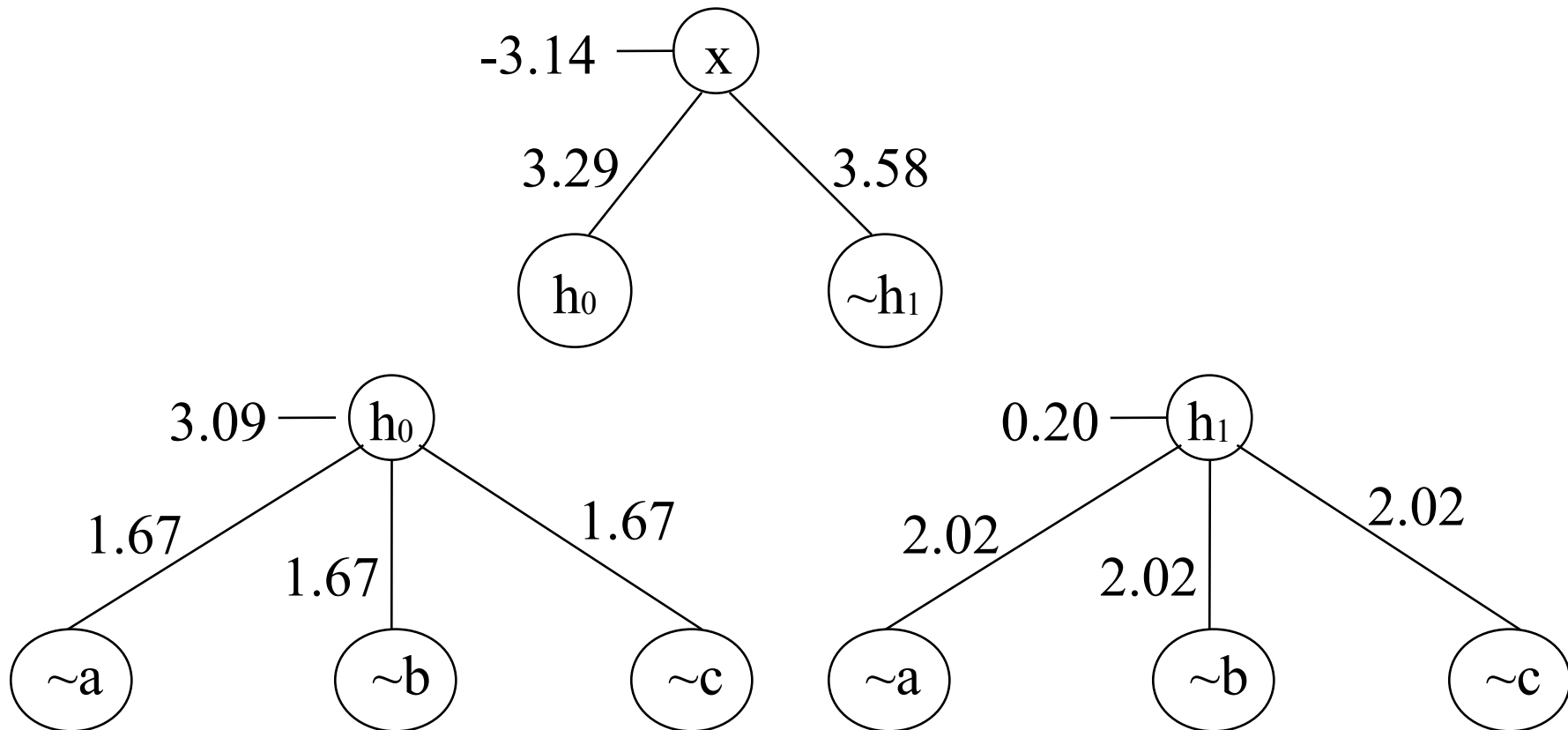


M of N Rule Simplification

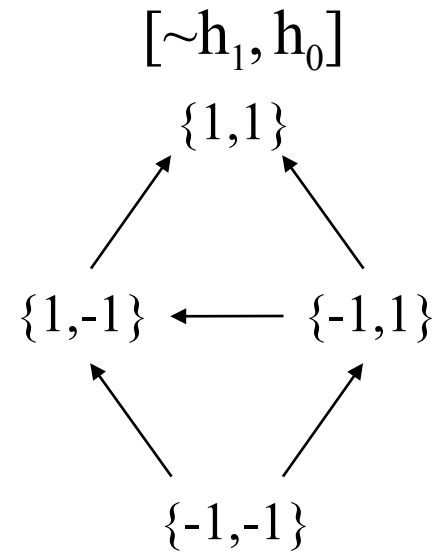
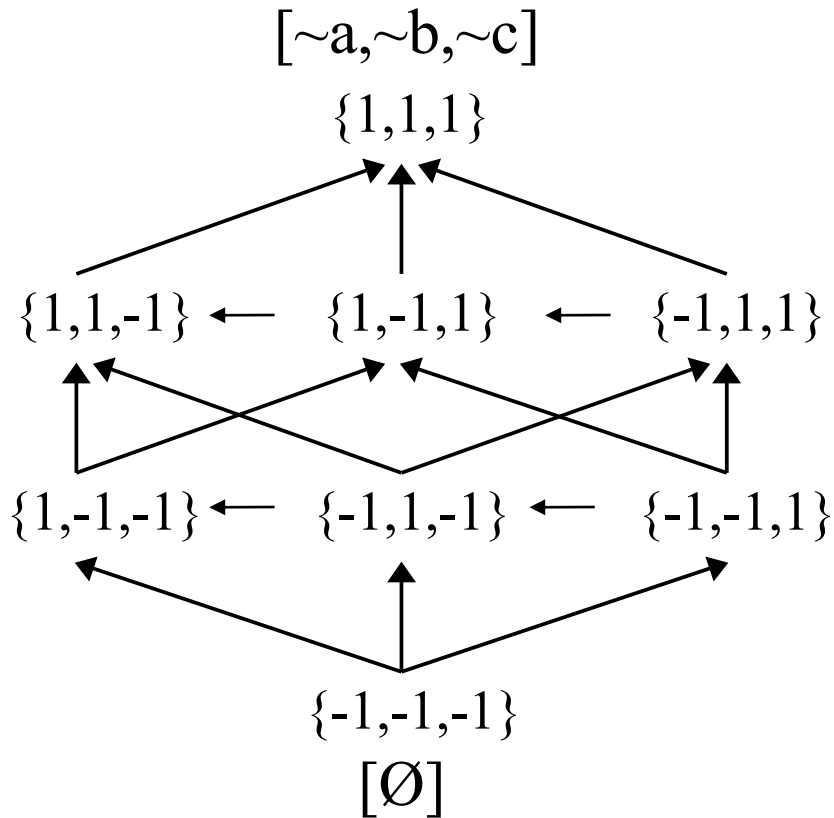


Sometimes it is difficult to find the partial ordering

- Step 1: Split network into sub-networks

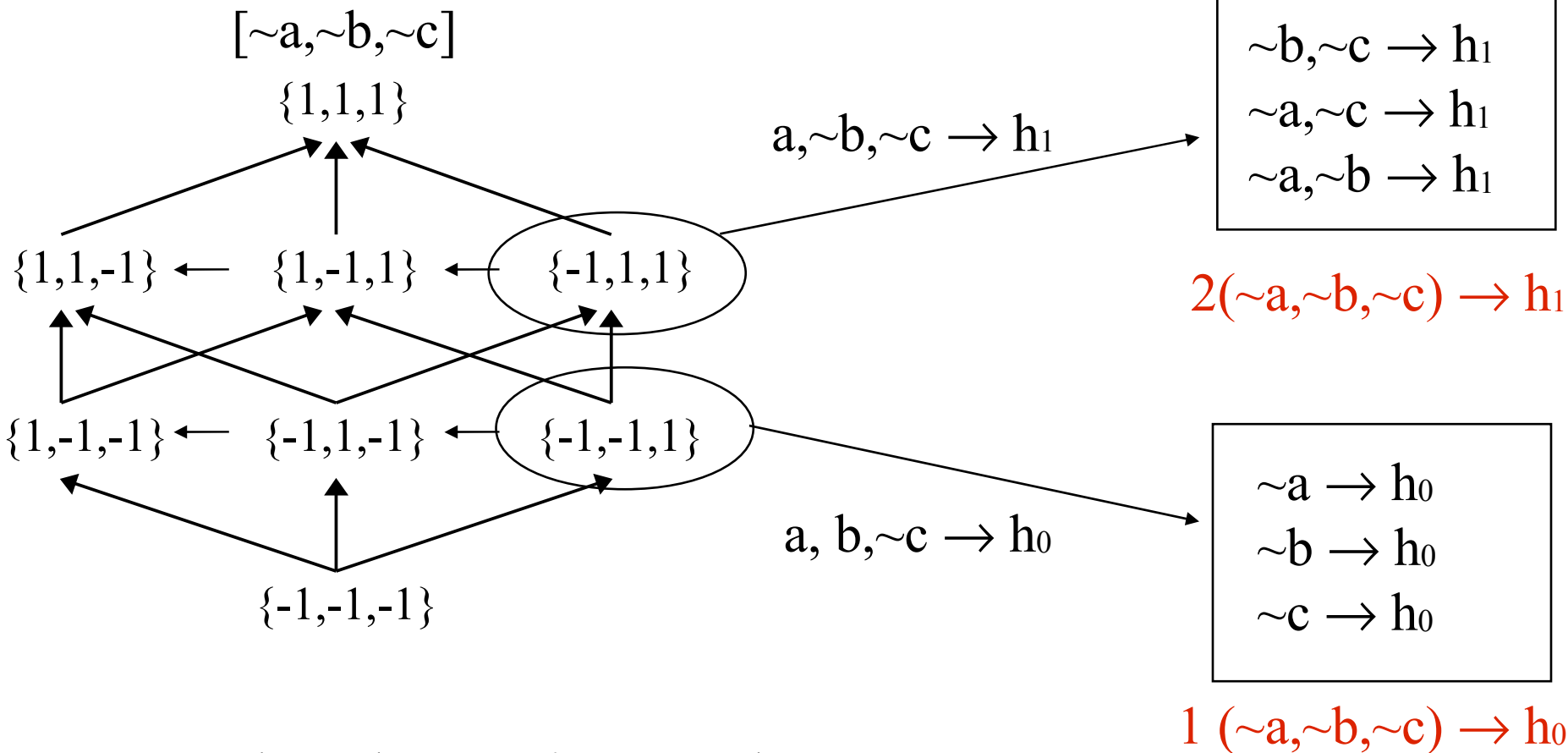


Step 2: Query the sub-networks



Step 3: Generate Rules (Input to Hidden)

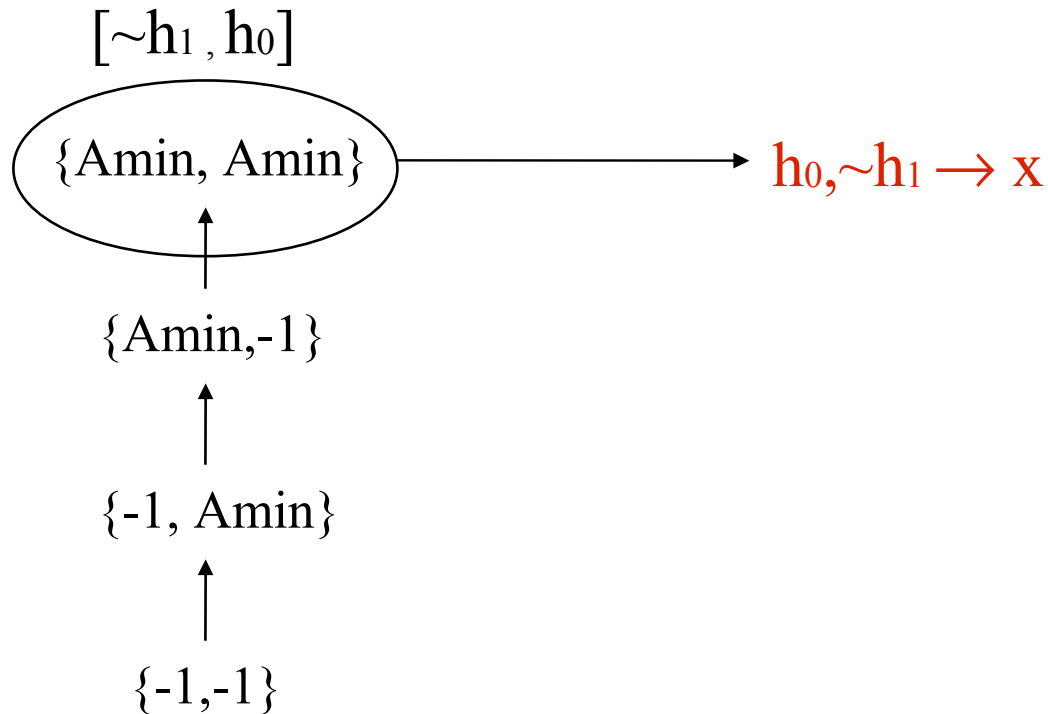
simplifying



We say that $a, b \rightarrow x$ *subsumes* $a, b, c \rightarrow x$

Step 4: Generate Rules (Hidden to Output)

- We have to take a conservative approach and use A_{min} , otherwise extraction can be unsound



Step 5: Merging the rules

- The Extraction of Rules from Input to Hidden layers is Sound and Complete (\rightarrow becomes \leftrightarrow)

$$1 (\sim a, \sim b, \sim c) \leftrightarrow h_0$$

$$2 (\sim a, \sim b, \sim c) \leftrightarrow h_1$$

$$h_0, \sim h_1 \rightarrow x$$

$$\longrightarrow 1(\sim a, \sim b, \sim c) \wedge \sim 2(\sim a, \sim b, \sim c) \rightarrow x$$

$$\downarrow 1(\sim a, \sim b, \sim c) \wedge 2(a, b, c) \rightarrow x$$

$$\downarrow \sim a, b, c \vee a, \sim b, c \vee a, b, \sim c \rightarrow x$$

(i.e. *exactly 2 out of $\{a, b, c\}$ imply x*)

The Extraction Algorithm is Sound

Summary

- The Extraction Method is Sound
- The Search Space is Reduced by Pruning Rules
- The Rule Set is Reduced by Simplification Rules

Future Work

- Language Extension:
 - Metalevel priorities
 - Extraction of first-order rules?
- Implementation:
 - Heuristics for searching the space of input vectors
 - MofN simplification during the generation of rules, instead of afterwards