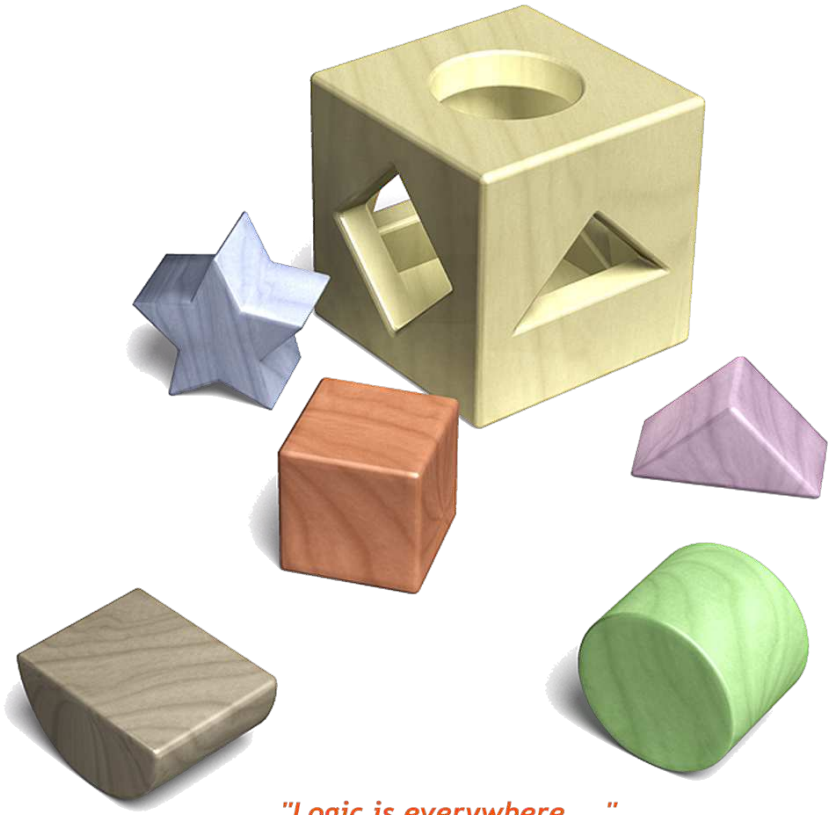
The Core Method

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- The Very Idea
- The Propositional CORE Method
- Human Reasoning



"Logic is everywhere ..."

The Very Idea

- **Various semantics for logic programs coincide with fixed points of associated immediate consequence operators** (Apt, van Emden: Contributions to the Theory of Logic Programming. Journal of the ACM 29, 841-862: 1982).
- **Banach Contraction Mapping Theorem** **A contraction mapping f defined on a complete metric space (X, d) has a unique fixed point. The sequence y, $f(y)$, $f(f(y))$, ... converges to this fixed point for any $y \in X$.**
  - **Consider programs whose immediate consequence operator is a contraction.** (Fitting: Metric Methods – Three Examples and a Theorem. Journal of Logic Programming 21, 113-127: 1994).
- **Every continuous function on the reals can be uniformly approximated by feed-forward connectionist networks** (Funahashi: On the Approximate Realization of Continuous Mappings by Neural Networks. Neural Networks 2, 183-192: 1989).
  - **Consider programs whose immediate consequence operator is continuous.** (H., Kalinke, Störr: Approximating the Semantics of Logic Programs by Recurrent Neural Networks. Applied Intelligence 11, 45-59: 1999).

First Ideas

- H., Kalinke: Towards a New Massively Parallel Computational Model for Logic Programming In: Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing, 68-77: 1994.

Interpretations

- Let $\mathcal{L}$ be a propositional language and $\{\top, \bot\}$ the set of truth values.
- An interpretation *I* is a mapping $\mathcal{L} \to \{\top, \bot\}$.
- For a given program $\mathcal{P}$, an interpretation *I* can be represented by the set of atoms occurring in $\mathcal{P}$ which are mapped to $\top$ under *I*, i.e. $I \subseteq \mathcal{R}_\mathcal{P}$.
- $2^{\mathcal{R}_\mathcal{P}}$ is the set of all interpretations for $\mathcal{P}$.
- $(2^{\mathcal{R}_\mathcal{P}}, \subseteq)$ is a complete lattice.
- An interpretation *I* for $\mathcal{P}$ is a model for $\mathcal{P}$ iff $I(\mathcal{P}) = \top$.

Immediate Consequence Operator

- **Immediate consequence operator $T_{\mathcal{P}} : 2^{\mathcal{R}_{\mathcal{P}}} \to 2^{\mathcal{R}_{\mathcal{P}}}$:**

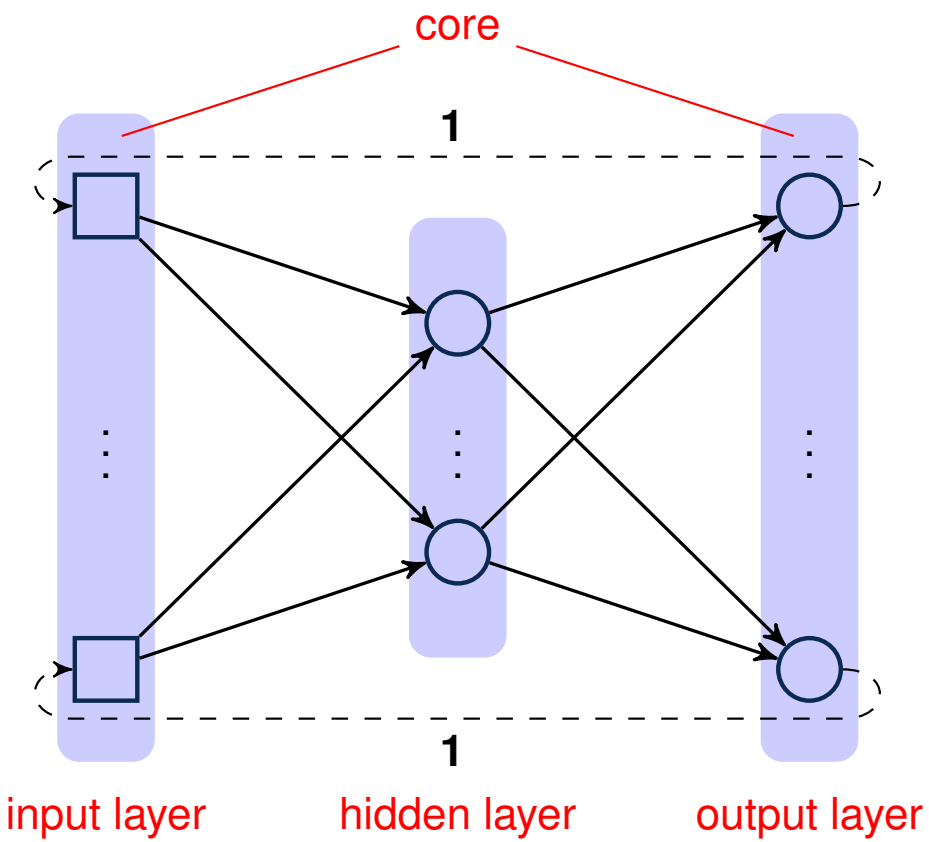
$$T_{\mathcal{P}}(I) = \{A \mid \text{there is a clause } A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{P} \text{ such that } I \models L_1 \wedge \ldots \wedge L_n\}.$$

- **I is a supported model iff $T_{\mathcal{P}}(I) = I$.**
- **Let lfp $T_{\mathcal{P}}$ be the least fixed point of $T_{\mathcal{P}}$ if it exists.**

The Propositional CORE Method

- Let $\mathcal{L}$ be a propositional logic language.
- Given a logic program $\mathcal{P}$ together with immediate consequence operator $T_{\mathcal{P}}$.
- Let $|\mathcal{R}_{\mathcal{P}}| = m$ and $2^{\mathcal{R}_{\mathcal{P}}}$ be the set of interpretations for $\mathcal{P}$.
- Find a mapping $\text{rep} : 2^{\mathcal{R}_{\mathcal{P}}} \to \mathbb{R}^m$.
- Construct a feed-forward network computing $f_{\mathcal{P}} : \mathbb{R}^m \to \mathbb{R}^m$, called the core, such that the following holds:
  - If $T_{\mathcal{P}}(I) = J$ then $f_{\mathcal{P}}(\text{rep}(I)) = \text{rep}(J)$, where $I, J \in 2^{\mathcal{R}_{\mathcal{P}}}$.
  - If $f_{\mathcal{P}}(\overline{s}) = \overline{t}$ then $T_{\mathcal{P}}(\text{rep}^{-1}(\overline{s})) = \text{rep}^{-1}(\overline{t})$, where $\overline{s}, \overline{t} \in \mathbb{R}^m$.
- Connect the units in the output layer recursively to the units in the input layer.
- Show that the following holds
  - $I = \text{lfp}\, T_{\mathcal{P}}$ iff the recurrent network converges to $\text{rep}(I)$, i.e. it reaches a stable state with input and output layer representing $\text{rep}(I)$.
- Connectionist model generation using recurrent networks with feed-forward core.

3-Layer Recurrent Networks



- **At each point in time all units do:**
  - **apply activation function to obtain potential,**
  - **apply output function to obtain output.**

Propositional CORE Method using Binary Threshold Units

- Let $\mathcal{L}$ be the language of propositional logic.
- Let $\mathcal{P}$ be a propositional logic program, e.g.,

$$\mathcal{P} = \{p,\ r \leftarrow p \wedge \sim q,\ r \leftarrow \sim p \wedge q\}.$$

- $T_{\mathcal{P}}(I) = \{A \mid A \leftarrow L_1 \wedge \ldots \wedge L_m \in \mathcal{P}$ such that $I \models L_1 \wedge \ldots \wedge L_m\}$.

$$\begin{array}{lcll} T_{\mathcal{P}}(\emptyset) & = & \{p\} & \\ T_{\mathcal{P}}(\{p\}) & = & \{p, r\} & \\ T_{\mathcal{P}}(\{p, r\}) & = & \{p, r\} & = \text{ lfp } T_{\mathcal{P}} \end{array}$$

Representing Interpretations

- $2^{\mathcal{R}_\mathcal{P}}$
- Let $m = |\mathcal{R}_\mathcal{P}|$ and identify $\mathcal{R}_\mathcal{P}$ with $\{1, \ldots, m\}$.
- Define rep : $2^{\mathcal{R}_\mathcal{P}} \to \mathbb{R}^m$ such that for all $1 \leq j \leq m$ we find:

$$\text{rep}(I)[j] = \begin{cases} 1 & \text{if } j \in I, \\ 0 & \text{if } j \notin I. \end{cases}$$

  E.g., if $\mathcal{R}_\mathcal{P} = \{p, q, r\} = \{1, 2, 3\}$ and $I = \{p, r\}$ then $\text{rep}(I) = (1, 0, 1)$.
- Other encodings are possible, e.g.,

$$\text{rep}'(I)[j] = \begin{cases} 1 & \text{if } j \in I, \\ -1 & \text{if } j \notin I. \end{cases}$$

- We can represent interpretations by arrays of binary or bipolar threshold units.

Computing the Core

- **Theorem** For each program $\mathcal{P}$,
  there exists a core of logical threshold units computing $T_{\mathcal{P}}$.
- **Proof** Let $\mathcal{P}$ be a program, $m = |\mathcal{R}_{\mathcal{P}}|$, and $\omega \in \mathbb{R}^+$.
  Wlog we assume that all occurrences of $\top$ in $\mathcal{P}$ have been eliminated.

  ▷ **Translation Algorithm**

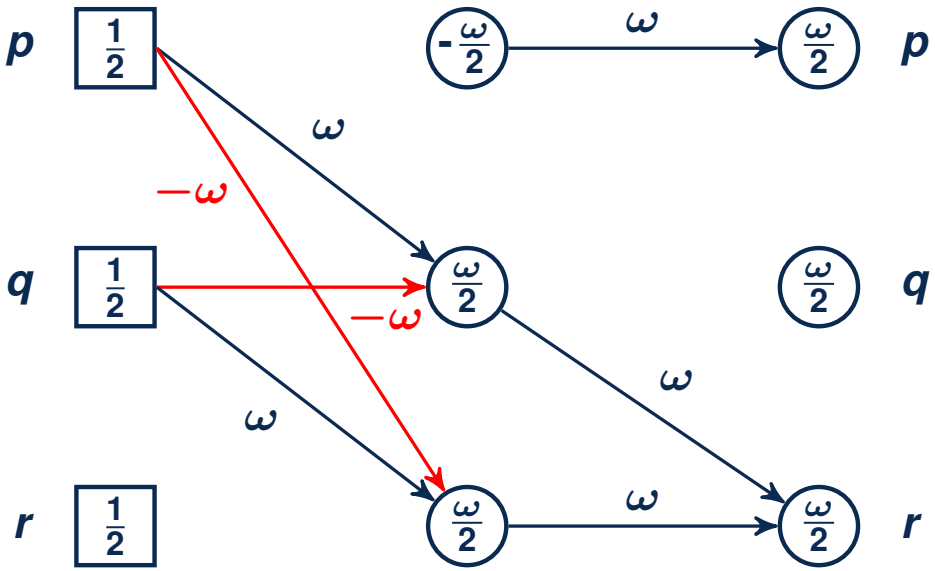
  1. Input and output layer: vector of length m of binary threshold units with threshold 0.5 and $\omega/2$ in the input and output layer, respectively.
  2. For each clause of the form $A \leftarrow L_1 \wedge \ldots \wedge L_k \in \mathcal{P}$, $k \geq 0$, do:
  - 2.1 Add a binary threshold unit u_h to the hidden layer.
  - 2.2 Connect u_h to the unit representing A in the output layer with weight ω.
  - 2.3 For each literal L_j, $1 \leq j \leq k$,
    connect the unit representing L_j in the input layer to u_h and,
    if L_j is an atom then set the weight to ω otherwise set the weight to $-\omega$.
  - 2.4 Set the threshold of u_h to $(p - 0.5)\omega$,
    where p is the number of positive literals occurring in $L_1 \wedge \ldots \wedge L_k$.

Computing the Core (Continued)

- **Theorem** For each program $\mathcal{P}$,
  there exists a core of logical threshold units computing $T_{\mathcal{P}}$.
- **Proof (Continued)** Some observations:
  - u_h becomes active at time $t+1$ iff $L_1 \wedge \ldots \wedge L_k$ is mapped to $\top$
    by the interpretation represented by the state of the input layer at time t.
  - The unit representing A in the output layer becomes active at time $t+2$
    iff there is a rule of the form $A \leftarrow L_1 \wedge \ldots \wedge L_k \in \mathcal{P}$ and
    the unit u_h in the hidden layer corresponding to this rule is active at time
    $t+1$.
  - The result follows immediately from these observations. qed

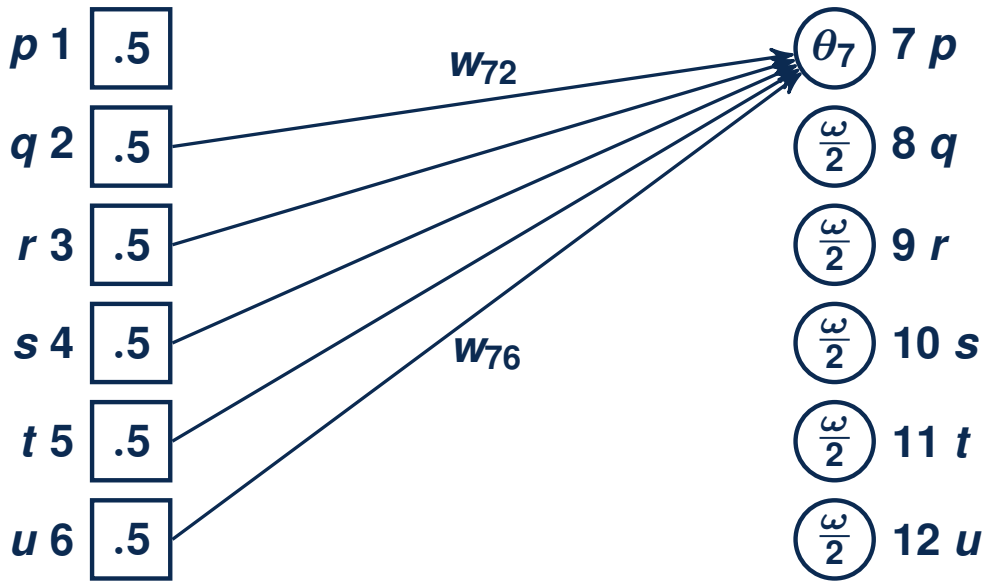
Computing the Core (Example)

- Consider again $\mathcal{P} = \{p,\ r \leftarrow p \wedge \sim q,\ r \leftarrow \sim p \wedge q\}$.
- The translation algorithm yields:



Hidden Layers are Needed

- The XOR can be represented by the program $\{r \leftarrow p \wedge \sim q,\ r \leftarrow \sim p \wedge q\}$.
- **Proposition** 2-layer networks cannot compute $T_{\mathcal{P}}$ for definite $\mathcal{P}$.
- **Proof** Suppose there exist 2-layer networks for definite $\mathcal{P}$ computing $T_{\mathcal{P}}$.
  - Consider $\mathcal{P} = \{p \leftarrow q,\ p \leftarrow r \wedge s,\ p \leftarrow t \wedge u\}$.



  - Let $\overline{v}$ be the state of the input layer; $\overline{v}$ is an interpretation.

Hidden Layers are Needed (Continued)

- **Proposition** 2-layer networks cannot compute $T_{\mathcal{P}}$ for definite $\mathcal{P}$.
- **Proof (Continued)** Consider $\mathcal{P} = \{p \leftarrow q,\ p \leftarrow r \wedge s,\ p \leftarrow t \wedge u\}$.
  - We have to find θ_7 and w_{7j}, $2 \leq j \leq 6$, such that

$$p \in T_{\mathcal{P}}(\overline{v}) \text{ iff } w_{72}v_2 + w_{73}v_3 + w_{74}v_4 + w_{75}v_5 + w_{76}v_6 \geq \theta_7.$$

  - Because conjunction is commutative we find

$$p \in T_{\mathcal{P}}(\overline{v}) \text{ iff } w_{72}v_2 + w_{74}v_3 + w_{73}v_4 + w_{76}v_5 + w_{75}v_6 \geq \theta_7.$$

  - Consequently,

$$p \in T_{\mathcal{P}}(\overline{v}) \text{ iff } w_{72}v_2 + w_1(v_3 + v_4) + w_2(v_5 + v_6) \geq \theta_7,$$

where $w_1 = \frac{1}{2}(w_{73} + w_{74})$ and $w_2 = \frac{1}{2}(w_{75} + w_{76})$.

Hidden Layers are Needed (Continued)

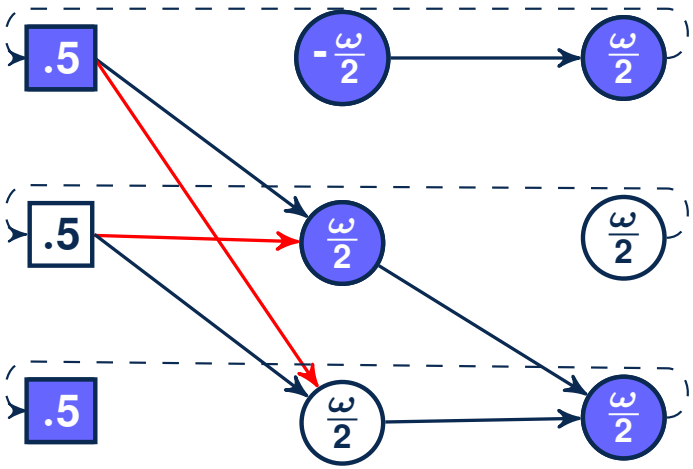
- **Proposition** 2-layer networks cannot compute $T_{\mathcal{P}}$ for definite $\mathcal{P}$.
- **Proof (Continued)** Consider $\mathcal{P} = \{p \leftarrow q,\ p \leftarrow r \wedge s,\ p \leftarrow t \wedge u\}$.
  - Likewise, because disjunction is commutative we find

$$p \in T_{\mathcal{P}}(\overline{v}) \text{ iff } w \cdot x \geq \theta_7,$$

    where $w = \frac{1}{3}(w_{72} + w_1 + w_2)$ and $x = \sum_{j=2}^{7} v_i$.
  - For the network to compute $T_{\mathcal{P}}$ the following must hold:
    - If $x = 0$ ($v_2 = \ldots = v_6 = 0$) then $w \cdot x - \theta_7 < 0$.
    - If $x = 1$ ($v_2 = 1,\ v_3 = \ldots = v_6 = 0$) then $w \cdot x - \theta_7 \geq 0$.
    - If $x = 2$ ($v_2 = v_4 = v_6 = 0,\ v_3 = v_5 = 1$) then $w \cdot x - \theta_7 < 0$.
  - However, $\frac{d(w \cdot x - \theta_7)}{dx} = w$ cannot change its sign; contradiction.
  - Consequently, 2-layer feed-forward networks cannot compute $T_{\mathcal{P}}$ qed

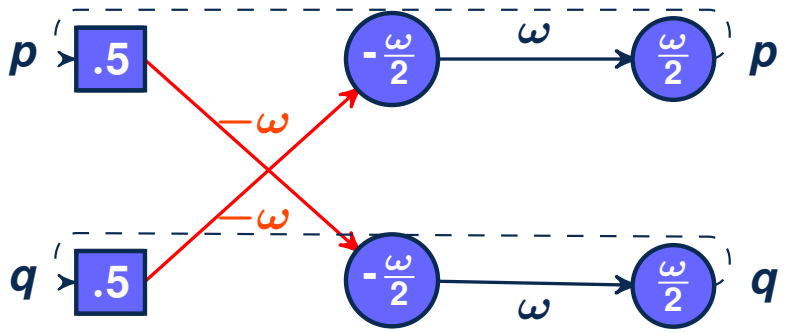
Adding Recurrent Connections

- **Recall** $\mathcal{P} = \{p,\ r \leftarrow p \wedge \sim q,\ r \leftarrow \sim p \wedge q\}$.



On the Existence of Least Fixed Points

- **Theorem** **For definite programs $\mathcal{P}$, $T_\mathcal{P}$ has a least fixed point which can be obtained by iterating $T_\mathcal{P}$ starting with the empty interpretation.** (Apt, van Emden: Contributions to the Theory of Logic Programming. Journal of the ACM 29, 841-862: 1982.)
- **In general, however, least fixed points do not always exist.**
  - **Consider $\mathcal{P} = \{p \leftarrow \sim q,\ q \leftarrow \sim p\}$**



  - **The corresponding recurrent network does not reach a stable state if initialized by the empty interpretation.**
  - **It has two stable states corresponding to the interpretations $\{p\}$ and $\{q\}$.**

Metrics for Logic Programs

- Let $\mathcal{P}$ be program and I a level mapping for $\mathcal{P}$.
- For interpretations $I, J \subseteq \mathcal{R}_\mathcal{P}$ we define

$$d_\mathcal{P} = \begin{cases} 0 & \text{if } I = J \\ 2^{-n} & \text{if } n \text{ is the smallest level on which } I \text{ and } J \text{ differ.} \end{cases}$$

- **Proposition 1** $(2^{\mathcal{R}_\mathcal{P}}, d_\mathcal{P})$ is a complete metric space.
- **Proposition 2** If $\mathcal{P}$ is acceptable, then there exists a metric space such that $T_\mathcal{P}$ is a contraction on it.
- For proofs of both Propositions see
  Fitting: Metric Methods – Three Examples and a Theorem.
  Journal of Logic Programming 21, 113-127: 1994.
- **Corollary** If $\mathcal{P}$ is acceptable, then there exists a 3-layer recurrent network of logical threshold units such that the computation starting with an arbitrary initial input converges and yields the unique fixed point of $T_\mathcal{P}$.

Time and Space Complexity

- Let $n = |\mathcal{P}|$ be the number of clauses
  and $m = |\mathcal{R}_m|$ be the number of propositional variables occurring in $\mathcal{P}$.
  - $2m + n$ units, $2mn$ connections in the core.
  - $T_{\mathcal{P}}(I)$ is computed in 2 steps.
  - The parallel computational model to compute $T_{\mathcal{P}}(I)$ is optimal.
    (A parallel computational model requiring $p(n)$ processors and $t(n)$ time to solve a problem of size n is *optimal* if $p(n) \cdot t(n) = O(T(n))$, where $T(n)$ is the sequential time to solve this problem; see: Karp, Ramachandran: Parallel Algorithms for Shared-Memory Machines. In: Handbook of Theoretical Computer Science, Elsevier, 869-941: 1990.)
  - The recurrent network settles down in $3n$ steps in the worst case.
    (Dowling, Gallier: Linear-time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. J. of Logic Programming 1, 267-284: 1984; Scutella: A Note on Dowling and Gallier's Top-Down Algorithm for Propositional Horn Satisfiability. J. of Logic Programming 8: 265-272: 1990.)
- **Exercise** Give an example of a program with worst case time behavior.

Reasoning wrt Least Fixed Points

- Let $\mathcal{P}$ be a program and assume that $T_{\mathcal{P}}$ admits a least fixed point.
- Let $\mathrm{lfp}\, T_{\mathcal{P}}$ denote the least fixed point of $\mathcal{P}$.
- It can be shown that $\mathrm{lfp}\, T_{\mathcal{P}}$ is the least model of $\mathcal{P}$.
- We define $\mathcal{P} \models^{\mathrm{lm}} F$ iff $\mathrm{lfp}\, T_{\mathcal{P}}(F) = \top$.
- Observe $\models^{\mathrm{lm}} \neq \models$.
- Consider $\mathcal{P} = \{p,\ q \leftarrow r\}$. Then,
  - $\mathrm{lfp}\, T_{\mathcal{P}} = \{p\}$ and
  - $\mathcal{P} \models^{\mathrm{lm}} p \wedge \sim q \wedge \sim r$, but
  - $\mathcal{P} \not\models \sim q$ and $\mathcal{P} \not\models \sim r$.
- If we consider $\models^{\mathrm{lm}}$, then negation is not classical negation.
  - This is the reason for using $\sim$ instead of $\neg$.
  - $\sim$ is often called negation by failure.

Extensions

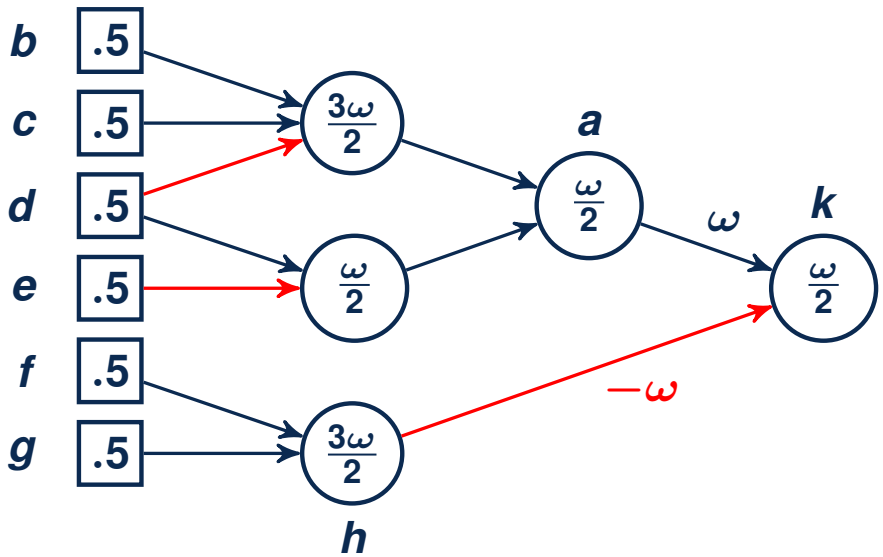
- **The approach has been extended to**
  - **many-valued logic programs** (Kalinke 1994, Seda, Lane 2004),
  - **extended logic programs** (d'Avila Garcez, Broda, Gabbay 2002),
  - **modal logic programs** (d'Avila Garcez, Lamb, Gabbay 2002),
  - **intuitionistic logic programs** d'Avila Garcez, Lamb, Gabbay 2003),
  - **first-order logic programs**
    (H., Kalinke, Störr 1999, Bader, Hitzler, H., Witzel 2007).

KBANN – Knowledge Based Artificial Neural Networks

- Towell, Shavlik: Extracting Refined Rules from Knowledge–Based Neural Networks. Machine Learning 131, 71-101: 1993.
  Can we do better than empirical learning?

- **Consider acyclic logic programs, e.g.,**

$$\mathcal{P} = \{a \leftarrow b \wedge c \wedge \sim d,\ a \leftarrow d \wedge \sim e,\ h \leftarrow f \wedge g,\ k \leftarrow a \wedge \sim h\}.$$

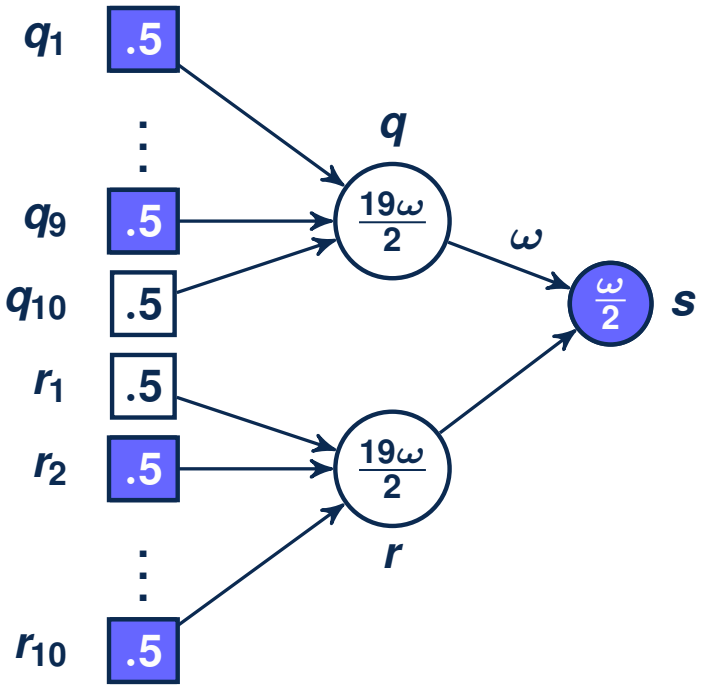


KBANN – Learning

- Given hierachical sets of propositional rules as background knowledge.
- Map rules into multi-layer feed-forward networks with sigmoidal units.
- Add hidden units (optional).
- Add units for known input features that are not referenced in the rules.
- Fully connect layers.
- Add near-zero random numbers to all links and thresholds.
- Apply backpropagation.
  - Empirical evaluation: system performs better than purely empirical and purely hand-built classifiers.

KBANN – A Problem

- Towel, Shavlik 1993: **"Works if rules have few conditions and there are few rules with the same head."**



- $p_q = p_r = 9\omega$ and $v_q = v_r = \frac{1}{1+e^{-\beta(9\omega-9.5\omega)}} \approx 0.46$ with $\beta = 1$.
- $p_s = 0.92\omega$ and $v_s = \frac{1}{1+e^{-\beta(0.92\omega-0.5\omega)}} \approx 0.6$ with $\beta = 1$.

Solving the Problem

- d'Avila Garcez, Zaverucha, Carvalho:
  Logic Programming and Inductive Learning in Artificial Neural Networks.
  In: Knowledge Representation in Neural Networks (Herrmann, Strohmaier, eds.), Logos, Berlin, 33-46: 1997.
  Can we combine the ideas of the propositational CORE method and KBANN while avoiding the above mentioned problem?
- **The approach has been generalized in**
  Bader: Neural-Symbolic Integration. PhD thesis, TU Dresden, Informatik: 2009.

Propositional CORE Method using Squashing Units

- Let u be a squashing unit.
  - Let $\Psi^- = \lim_{p \to -\infty} \Psi(p)$ and $\Psi^+ = \lim_{p \to \infty} \Psi(p)$.
  - Let $a^-, a^0, a^+ \in \mathbb{R}$ such that $\Psi^- < a^- < a^0 < a^+ < \Psi^+$.
  - u is active iff $v \geq a^+$.
    u is passive iff $v \leq a^-$.
    u is in null-state iff $v = a^0$.
    u is undecided iff $a^- < (v \neq a^0) < a^+$.
  - $p^+ = \Psi^{-1}(a^+)$ is called minimal activation potential.
    $p^- = \Psi^{-1}(a^-)$ is called maximal inactivation potential.
    $p^0 = \Psi^{-1}(a^0)$ is called null-state potential.

The Task

- **How can we guarantee that a unit is either active, passive or in the null-state?**
- **Suppose input layer units output only finitely many values.**
- **Let u be a hidden layer unit.**
- **If the input layer is finite,**
  then the potential of u may only take finitely many different values.
- **Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the set of possible values for the potential of u.**
- **Let $\mathcal{P}^+ = \{p \in \mathcal{P} \mid p > p^0\}$ and $\mathcal{P}^- = \{p \in \mathcal{P} \mid p < p^0\}$.**
- **Let $m = \max(m^-, m^+)$, where**

$$m^+ = \left\{ \begin{array}{ll} 0 & \text{if } \mathcal{P}^+ = \emptyset \\ \left|\frac{p^+}{\min(\mathcal{P}^+) - p^0}\right| & \text{otherwise} \end{array} \right\}, \quad m^- = \left\{ \begin{array}{ll} 0 & \text{if } \mathcal{P}^- = \emptyset \\ \left|\frac{p^-}{p^0 - \min(\mathcal{P}^-)}\right| & \text{otherwise} \end{array} \right\}.$$

- **Observations**
  - **If the weights on the connections to u and the threshold of u are multiplied with m, then u is either active, passive or in the null-state.**
  - **u produces only finitely many different output values.**
  - **The transformation can be applied to output layer units as well.**

Example

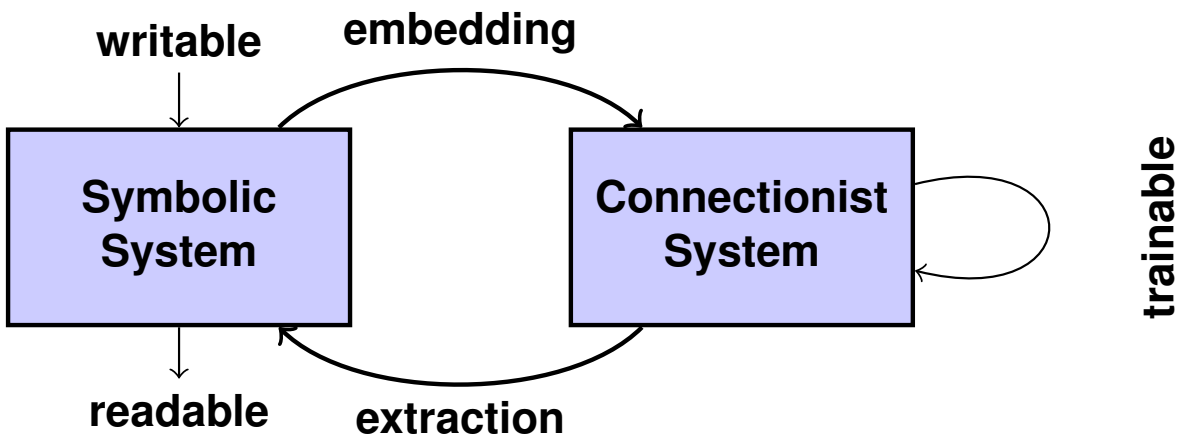
- Consider a bipolar sigmoidal unit u with $\Psi(p) = \tanh(p)$.
- Let $\mathcal{P} = \{-0.9, -0.5, -0.3, 0.0, 0.2, 0.4, 0.8\}$ and $a^- = -0.8$, $a^0 = 0.0$, $a^+ = 0.8$.
- Then, $p^- = \Psi^{-1}(a^-) \approx -1.1$, $p^0 = \Psi^{-1}(a^0) = 0.0$, $p^+ = \Psi^{-1}(a^+) \approx 1.1$.
- Hence, $m^- = |\frac{-1.1}{0.0+0.3}| = 3.662$ and $m^+ = |\frac{1.1}{0.2-0.0}| = 5.493$.
- Thus, $m = \max(m^-, m^+) = \max(3.662, 5.493) = 5.493$ and we obtain

p	tanh(p)	tanh(m × p)
-0.9	-0.716	-0.999
-0.5	-0.462	-0.992
-0.3	-0.291	-0.929
0.0	0.000	0.000
0.2	0.197	0.800
0.4	0.380	0.976
0.8	0.664	0.999

- **Exercise** Specify a core of bipolar sigmoidal units for $\mathcal{P} = \{p,\ r \leftarrow p \wedge \sim q,\ r \leftarrow \sim p \wedge q\}$ with $a^- = -0.9$, $a^0 = 0.0$, $a^+ = 0.9$.

Results

- Relation to logic programs is preserved.
  - For each program $\mathcal{P}$, there exists a core of squashing units computing $T_{\mathcal{P}}$.
  - If $\mathcal{P}$ is acceptable, then there exists a 3-layer recurrent network of squashing units such that the computation starting with an arbitrary initial input converges and yields the unique fixed point of $T_{\mathcal{P}}$.
  - Likewise for consistent acceptable extended logic programs.
- The core is trainable by backpropagation.
- The neural symbolic cycle

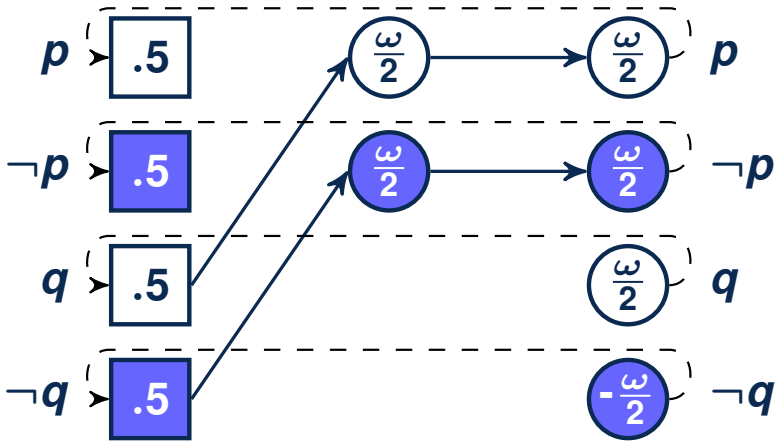


Cores for Three-Valued Logic Programs

- Consider $\{p \leftarrow q\}$.
- A translation algorithm translates programs into a core.
- Recurrent connections connect the output to the input layer.

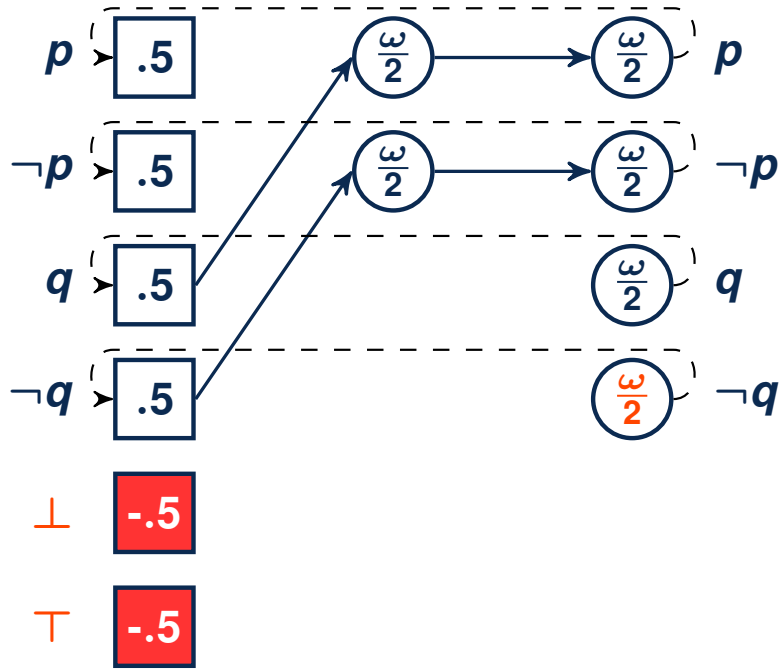
Kalinke 1995, Seda, Lane 2004

Φ_F



new

Φ_{SvL}

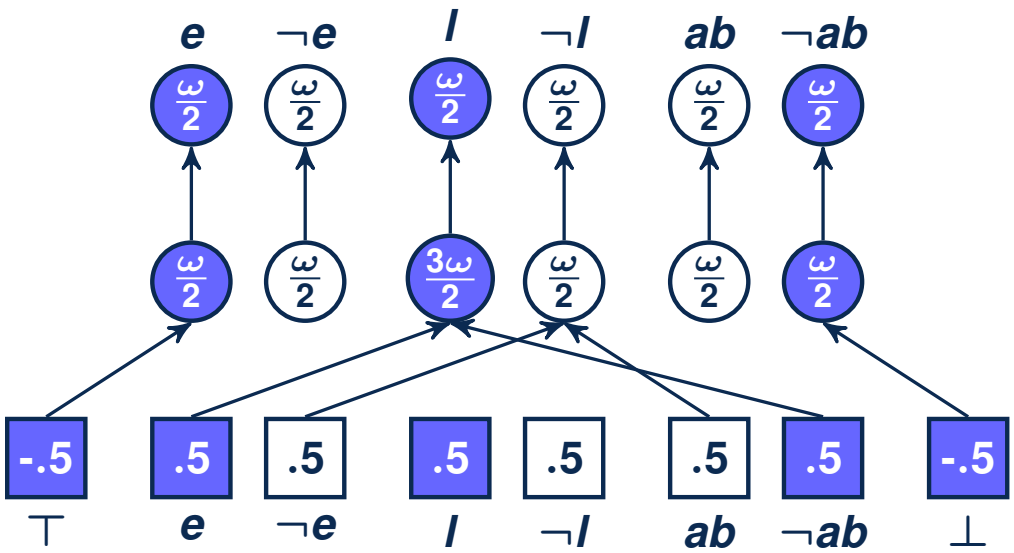


A CORE Method for Human Reasoning

- Consider three-layer feed forward networks of binary threshold units.
- The input as well as the output layer shall represent interpretations.
- **Theorem**
  For each program $\mathcal{P}$ there exists a feed-forward core computing $\Phi_{\mathsf{SvL},\mathcal{P}}$.
- Add recurrent connections between corresponding units
  in the output and the input layer.
- **Corollary**
  The recurrent network reaches a stable state representing lfp $\Phi_{\mathsf{SvL},\mathcal{P}}$
  if initialized with $\langle \emptyset, \emptyset \rangle$.

The Suppression Task – Modus Ponens

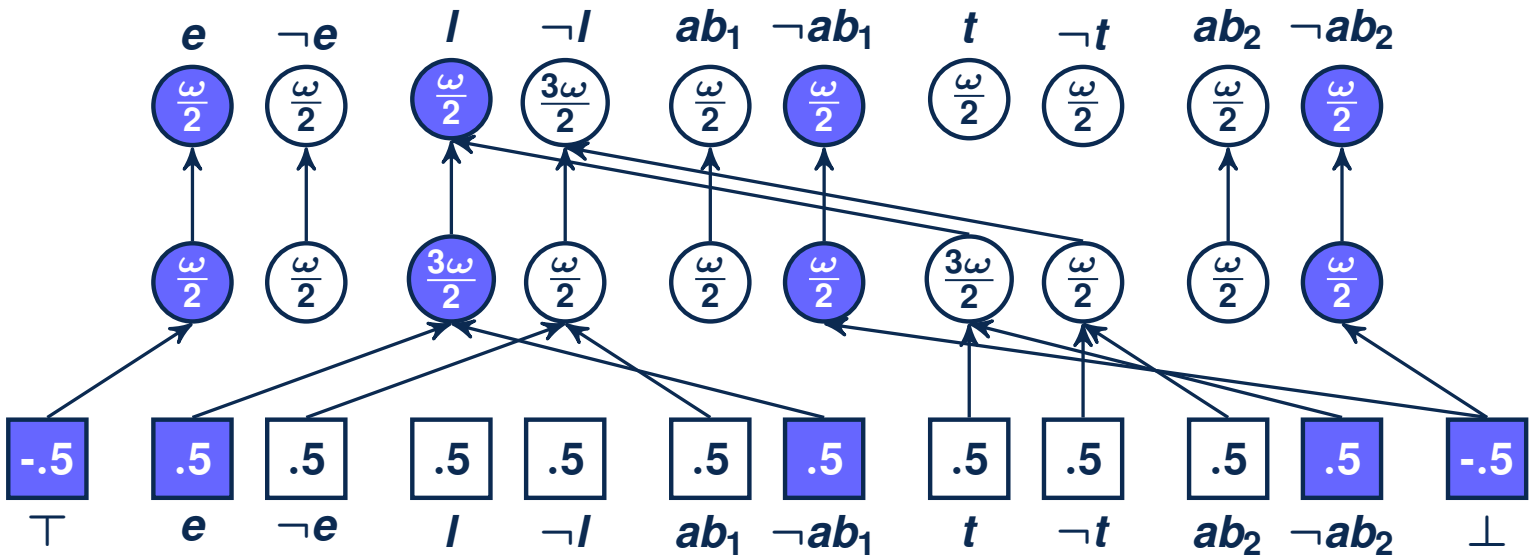
- *If she has an essay to write, she will study late in the library.*
  She has an essay to write.
- **96% of subjects conclude that she will study late in the library.**
- $\mathcal{P}_4 = \{I \leftarrow e \wedge \neg ab,\ e \leftarrow \top,\ ab \leftarrow \bot\}$.



- $\mathrm{lfp}\,\Phi_{\mathrm{SvL},\mathcal{P}_4} = \mathrm{lm}_{3Ł}\,\mathrm{wc}\,\mathcal{P}_4 = \langle\{I, e\}, \{ab\}\rangle$.
- **From $\langle\{I, e\}, \{ab\}\rangle$ follows that she will study late in the library.**

The Suppression Task – Alternative Arguments

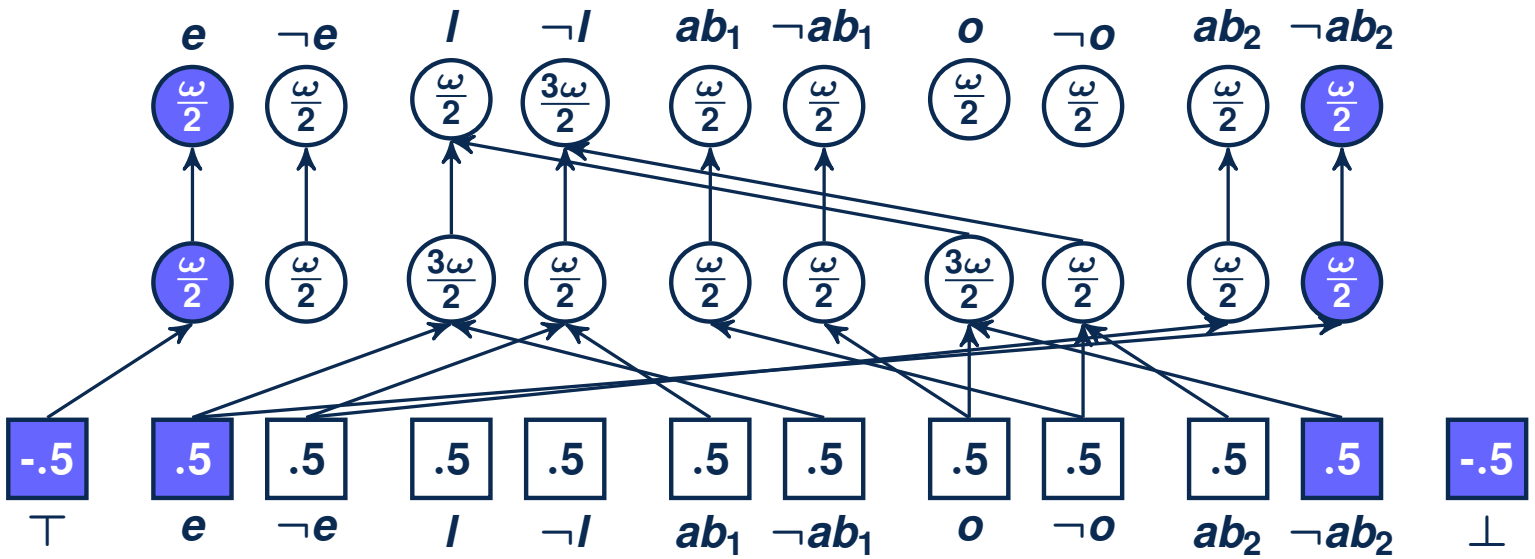
- *If she has an essay to write, she will study late in the library. She has an essay to write. If she has some textbooks to read, she will study late in the library.*
- **96% of subjects conclude that she will study late in the library.**
- $\mathcal{P}_5 = \{ I \leftarrow e \wedge \neg ab_1,\ e \leftarrow \top,\ ab_1 \leftarrow \bot,\ I \leftarrow t \wedge \neg ab_2,\ ab_2 \leftarrow \bot \}$.



- $\mathrm{lfp}\, \Phi_{\mathrm{SvL},\mathcal{P}_5} = \mathrm{lm}_{3Ł}\, \mathrm{wc}\, \mathcal{P}_5 = \langle \{e, I\}, \{ab_1, ab_2\} \rangle$.
- **From $\langle \{e, I\}, \{ab_1, ab_2\} \rangle$ follows that she will study late in the library.**

The Suppression Task – Additional Argument

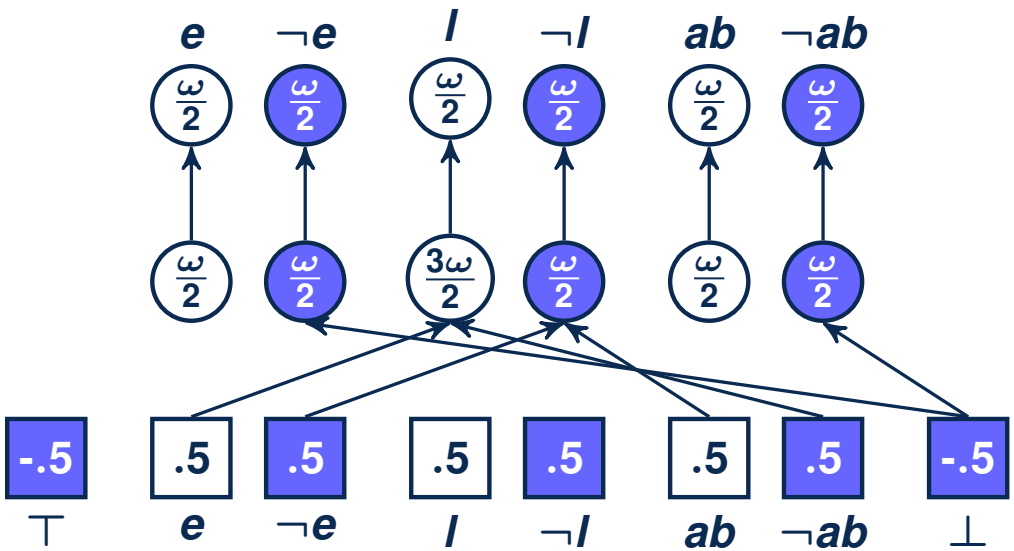
- *If she has an essay to write, she will study late in the library. She has an essay to write. If the library stays open, she will study late in the library.*
- **38% of subjects conclude that she will study late in the library.**
- $\mathcal{P}_6 = \{l \leftarrow e \wedge \neg ab_1,\ e \leftarrow \top,\ l \leftarrow o \wedge \neg ab_2,\ ab_1 \leftarrow \neg o,\ ab_2 \leftarrow \neg e,\}$.



- $\mathrm{lfp}\,\Phi_{\mathrm{SvL},\mathcal{P}_6} = \mathrm{Im}_{3Ł}\,\mathrm{wc}\,\mathcal{P}_6 = \langle\{e\},\{ab_2\}\rangle$.
- **From $\langle\{e\},\{ab_2\}\rangle$ follows that it is unknown whether she will study late in the library.**

The Suppression Task – Denial of Antecendent (DA)

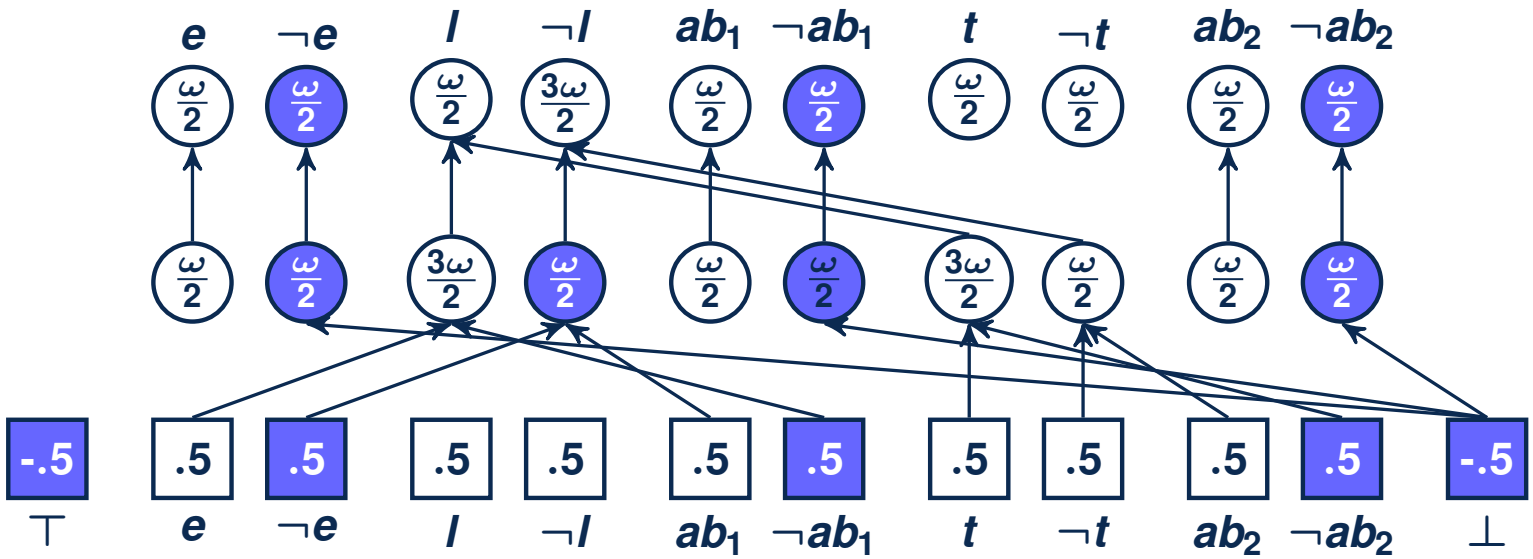
- *If she has an essay to write, she will study late in the library.*
  She does not have an essay to write.
- **46% of subjects conclude that she will not study late in the library.**
- $\mathcal{P}_7 = \{l \leftarrow e \wedge \neg ab,\ e \leftarrow \bot, ab \leftarrow \bot\}$.



- $\text{lfp}\ \Phi_{\text{SvL},\mathcal{P}_7} = \text{lm}_{3Ł}\ \text{wc}\ \mathcal{P}_7 = \langle \emptyset, \{ab, e, l\}\rangle$.
- **From $\langle \emptyset, \{ab, e, l\}\rangle$ follows that she will not study late in the library.**

The Suppression Task – Alternative Argument and DA

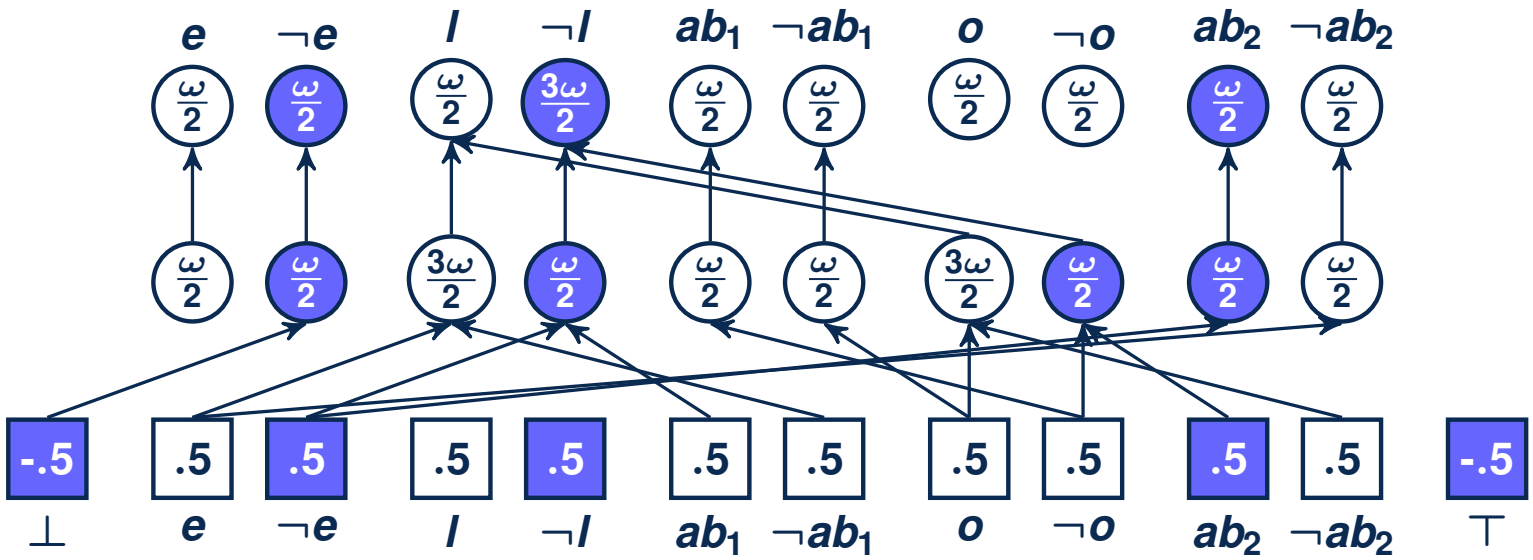
- *If she has an essay to write, she will study late in the library. She does not have an essay to write. If she has textbooks to read, she will study late in the library.*
- **4% of subjects conclude that Marian will not study late in the library.**
- $\mathcal{P}_8 = \{l \leftarrow e \wedge \neg ab_1,\ e \leftarrow \bot,\ ab_1 \leftarrow \bot,\ l \leftarrow t \wedge \neg ab_2,\ ab_2 \leftarrow \bot\}$.



- $\mathrm{lfp}\,\Phi_{\mathrm{SvL},\mathcal{P}_8} = \mathrm{Im}_{3Ł}\,\mathrm{wc}\,\mathcal{P}_8 = \langle \emptyset, \{ab_1, ab_2, e\}\rangle$.
- **From $\langle \emptyset, \{ab_1, ab_2, e\}\rangle$ follows that it is unknown whether she will study late in the library.**

The Suppression Task – Additional Argument and DA

- *If she has an essay to write, she will study late in the library. She does not have an essay to write. If the library is open, she will study late in the library.*
- **63% of subjects conclude that she will not study late in the library.**
- $\mathcal{P}_9 = \{l \leftarrow e \wedge \neg ab_1,\ e \leftarrow \bot,\ l \leftarrow o \wedge \neg ab_2,\ ab_1 \leftarrow \neg o,\ ab_2 \leftarrow \neg e\}$.



- $\mathrm{lfp}\,\Phi_{\mathrm{SvL},\mathcal{P}_9} = \mathrm{Im}_{3Ł}\,\mathrm{wc}\,\mathcal{P}_9 = \langle\{ab_2\}, \{e, l\}\rangle$.
- **From $\langle\{ab_2\}, \{e, l\}\rangle$ follows that she will not study late in the library.**

Summary

- Under Łukasiewicz semantics we obtain

	Byrne 1989	Program	$\mathrm{Im}_{3Ł}\ \mathrm{wc}\ \mathcal{P}_i(I)$
Modus Ponens	I (96%)	$\mathcal{P}_4$	$\top$
Alternative Arguments	I (96%)	$\mathcal{P}_5$	$\top$
Additional Arguments	I (38%)	$\mathcal{P}_6$	U
Modus Ponens and DA	$\neg I$ (46%)	$\mathcal{P}_7$	$\bot$
Alternative Arguments and DA	$\neg I$ (4%)	$\mathcal{P}_8$	U
Additional Arguments and DA	$\neg I$ (63%)	$\mathcal{P}_9$	$\bot$

- The approach appears to be adequate.
- Fitting semantics or completion is inadequate.