Computational Logic and Connectionist Systems

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- Introduction
- Propositional Logic
- First-Order Logic
- Application
- Challenge Problems

"Logic is everywhere ..."
Motivation: Conjunction Search (1)

- Treisman, Sato 1990.
- Where is the triangle?
- ?- triangle(X).
- triangle(a).
  square(b).
  square(c).
  :
Motivation: Conjunction Search (2)

- Where is the blue triangle?
- \(?- \text{triangle}(X), \text{blue}(X).\)
- \(\text{triangle}(a).\)
  \(\text{triangle}(b)\)
  \(\vdash\)
  \(\text{square}(c).\)
  \(\text{square}(d).\)
  \(\vdash\)
  \(\text{blue}(b).\)
  \(\text{blue}(c).\)
  \(\vdash\)
  \(\text{red}(a).\)
  \(\text{red}(d).\)
  \(\vdash\)
Motivation: Speech Understanding and Spatial Reasoning

▶ “A large circle is above a dark square”

▶ ?- obj(o(X,circle,_,large)),
   obj(o(Y,square,dark,_)),
   above(X,Y).

▶ above(Trajector,Landmark) :-
   rAbove(Landmark,RegionAbove),
   in(Trajector,RegionAbove).

▶ obj(o(a,circle,light,large)).
   obj(o(b,square,dark,medium)).

▶ ?- rAbove(b,RegionAbove),
   in(RegionAbove,a).

▶ Beringer, Hölldobler, Kurfeß 1993
Introduction: Connectionist Systems

- Well-suited to learn, to adapt to new environments, to degrade gracefully etc.
- Many successful applications.
- Approximate functions.
  - Hardly any knowledge about the functions is needed.
  - Trained using incomplete data.
- A declarative semantics is not available.
- Recursive networks are hardly understood.
- **McCarthy 1988**: We still observe a propositional fixation.
- Structured objects are difficult to represent and to reason about.
  - **Smolensky 1987**: Can we instantiate the power of symbolic computation within fully connectionist systems?
Introduction: Computational Logic

- Well-suited to represent and reason about structured objects and structure-sensitive processes.
- Many successful applications.
- Direct implementation of relations and functions.
- Explicit expert knowledge is required,
  - but may not be available.
- Highly recursive structures.
- Well-understood declarative semantics.
- Logic systems are brittle.

- Can we instantiate the power of connectionist computation within a computational logic system?
Introduction: Objective

- Seek the best of both paradigms!
- Understanding the relation between connectionist and logic systems.
- Contribute to the open research problems of both areas.
- Well-developed for propositional case.
- Hard problem: going beyond.

In this lecture:

- (Partial) overview on existing approaches.
- Logic programs and recurrent networks.
- **Semantic operators for logic programs can be computed by connectionist systems.**
- Semantic operators can be learned.
- Logic programs can be extracted.

**Neural Symbolic Integration using the Core Method**
Introduction: Connectionist Networks

- A connectionist network consists of
  - a set \( U \) of units and
  - a set \( W \subseteq U \times U \) of connections.

- Each connection is labeled by a weight \( w \in \mathbb{R} \).

- If there is a connection from unit \( u_j \) to \( u_k \), then \( w_{kj} \) is its associated weight.

- A unit is specified by
  - an input vector \( \vec{i} = (i_1, \ldots, i_m) \), \( i_j \in \mathbb{R} \), \( 1 \leq j \leq m \),
  - an activation function \( \Phi \) mapping \( \vec{i} \) to a potential \( p \in \mathbb{R} \),
  - an output function \( \Psi \) mapping \( p \) to an (output) value \( v \in \mathbb{R} \).

- If there is a connection from \( u_j \) to \( u_k \)
  then \( w_{kj} v_j \) is the input received by \( u_k \) along this connection.

- In most networks the potential and value of a unit are synchronously updated.

- Often a linear time \( t \) is added as parameter to input, potential and value.

- The state of a network with units \( u_1, \ldots, u_n \) at time \( t \) is \( (v_1(t), \ldots, v_n(t)) \).
Literature

Propositional Logic

► Some Existing Approaches
  ▶ Finite Automata and McCulloch-Pitts Networks
  ▶ Propositional Reasoning and Symmetric/Stochastic Networks

► Propositional Logic Programs and the Core Method
  ▶ The Neural-Symbolic Cycle
  ▶ The Very Idea
  ▶ Logic Programs
  ▶ Propositional Core Method
  ▶ Backpropagation
  ▶ Knowledge-Based Artificial Neural Networks
  ▶ Propositional Core Method using Sigmoidal Units
  ▶ Further Extensions
McCulloch-Pitts Networks

- McCulloch, Pitts 1943: Can the activities of nervous systems be modelled by a logical calculus?
- A McCulloch-Pitts network consists of a finite set $U$ of binary threshold units and a set $W \subseteq U \times U$ of weighted connections.
- The set $U_I$ of input units is defined as $U_I = \{ u_k \in U \mid (\forall u_j \in U) \ w_{kj} = 0 \}$.
- The set $U_O$ of output units is defined as $U_O = \{ u_j \in U \mid (\forall u_k \in U) \ w_{kj} = 0 \}$.
Binary Threshold Units

- \( u_k \) is a binary threshold unit if

\[
\Phi(\vec{i}_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j
\]
\[
\Psi(p_k) = v_k = \begin{cases} 
1 & \text{if } p_k \geq \theta_k \\
0 & \text{otherwise}
\end{cases}
\]

where \( \theta_k \in \mathbb{R} \) is a threshold.

- Three binary threshold units:

1. \( u_2 \)
   - \( v_1 \)
   - \( w_{21} = -1 \)
   - \( \theta_2 = -0.5 \)
   - \( v_2 = \neg v_1 \)

2. \( u_3 \)
   - \( v_1 \)
   - \( w_{31} = 1 \)
   - \( \theta_3 = 0.5 \)
   - \( v_3 = v_1 \lor v_2 \)

3. \( u_3 \)
   - \( v_2 \)
   - \( w_{32} = 1 \)
   - \( \theta_3 = 1.5 \)
   - \( v_3 = v_1 \land v_2 \)
A Simple McCulloch-Pitts Network

Example Consider the following network of logical threshold units:

Exercise

What is computed by the network if all units are updated synchronously?
Specify the states of the network ignoring input and output units.
Finite Automata

A finite automaton consists of:

- $\Sigma$, a finite set of input symbols,
- $\Phi$, a finite set of output symbols,
- $Q$, a finite set of states,
- $q_0 \in Q$, an initial state,
- $F \subset Q$, a set of final states,
- $\delta : Q \times \Sigma \rightarrow Q$, a state transition function,
- $\rho : Q \rightarrow \Phi$, an output function.

Exercise Let $\Sigma = \Phi = \{1, 2\}$, $Q = \{p, q, r\}$, $F = \{r\}$, $q_0 = p$,

\[
\begin{array}{c|ccc}
\rho & p & q & r \\
\hline
1 & 1 & 2 \\
\end{array} \quad \begin{array}{c|cc}
\delta & 1 & 2 \\
\hline
p & q & p \\
q & r & q \\
r & r & r \\
\end{array}
\]

What is computed by this automaton?
Finite Automata and McCulloch-Pitts Networks

Theorem McCulloch-Pitts networks are finite automata and vice versa.

Proof

Exercise

Let $T = (\Sigma, \Phi, Q, q_0, F, \delta, \rho)$ an automaton with

- $\Sigma = \{b_1, \ldots, b_m\}$,
- $\Phi = \{c_1, \ldots, c_r\}$,
- $Q = \{q_0, \ldots, q_{k-1}\}$.

To show there exists network $N$ with

- inputs $\{b'_1, \ldots, b'_m\}$,
- outputs $\{c'_1, \ldots, c'_r\}$,
- states $\{q'_0, \ldots, q'_{k-1}\}$ such that

if $T$ generates $c_{j_1}, \ldots, c_{j_n}$ given $b_{j_1}, \ldots, b_{j_n}$
then $N$ generates $c'_{j_1}, \ldots, c'_{j_n}$ given $b'_{j_1}, \ldots, b'_{j_n}$. 
Construction of the Network: Inputs and Outputs

- Remember $|\Sigma| = m, |\Phi| = r$.

- Inputs $x_1, \ldots, x_m$ with $b'_j = \vec{x}$ where

  $$x_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

- Outputs $y_1, \ldots, y_r$ with $c'_j = \vec{y}$ where

  $$y_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$
Construction of the Network: Units and Connections

▶ Remember $|\Sigma| = m$, $|\Phi| = r$, $|Q| = k$.

▶ $qb$-units represent that $T$ in state $q$ receives input $b$ ($k \times m$ units).

▶ $c$-units represent output $c$ ($r$ units).

▶ Connections

▷ Let $\{k_1, \ldots, k_{n(k)}\} = \{(q, b) \mid \delta(q, b) = q^*\}$ in

$$v_{u_q b^*}(t + 1) = \begin{cases} 1 & \text{if } x_{b^*}(t) \land [k_1(t) \lor \ldots \lor k_{n(k)}(t)], \\ 0 & \text{otherwise}. \end{cases}$$

▷ Let $\{l_1, \ldots, l_{n(l)}\} = \{(q, b) \mid \rho(q) = c\}$ in

$$v_{u_c}(t + 1) = \begin{cases} 1 & \text{if } l_1(t) \lor \ldots \lor l_{n(l)}(t), \\ 0 & \text{otherwise}. \end{cases}$$

▶ The theorem follows by induction on the length of the input sequence.
Exercises

- Specify the automaton corresponding to the sample network.
- Specify the network corresponding to the sample finite automaton.
- Complete the proof of the theorem.
Some Remarks on McCulloch-Pitts Networks

- McCulloch-Pitts networks are not just simple reactive systems, but their behavior depends on previous inputs as well as the activity within the network.

Example

- Some of the results by McCulloch and Pitts can be extended to weighted automata Bader, Hölldobler, Scalzitti 2004.
Literature

Symmetric Networks

- Hopfield 1982: Can statistical models for magnetic materials explain the behavior of certain classes of networks?

- Original application: associative memory.

- A symmetric network consists of a finite set $U$ of binary threshold units and a set $W \subseteq U \times U$ of weighted connections such that $w_{kj} = w_{jk}$ and $w_{kk} = 0$ for all $k, j$ with $k \neq j$.

- Asynchronous update procedure: while state $\vec{v}$ is unstable: update an arbitrary unit.
Energy Minimization

► What happens precisely when a symmetric network is updated?

► Consider the energy function

\[
E(t) = -\frac{1}{2} \sum_{k,j} w_{kj} v_j(t) v_k(t) + \sum_k \theta_k v_k(t)
\]

\[
= -\sum_{k<j} w_{kj} v_j(t) v_k(t) + \sum_k \theta_k v_k(t)
\]

describing the state of the network at time \(t\).

► Remember \(w_{kk} = 0\) for all \(k\).

► Exercise

▷ Specify \(E(t)\) for the symmetric networks on the previous page.

▷ How does an update change the energy of a symmetric network (you may assume that \(\theta_k = 0\) for all \(k\))? 

► Theorem \(E\) is monotone decreasing, i.e., \(E(t+1) \leq E(t)\).

► Exercise Does this theorem still hold if we drop the assumption that \(w_{ij} = w_{ji}\)?

► Exercise How plausible is the assumption that \(w_{ij} = w_{ji}\)?
Stochastic Networks or Boltzmann Machines

- Hinton, Sejnowski 1983: Can we escape local minima?
- A stochastic network is a symmetric network, but the values are computed probabilistically

\[ P(v_k = 1) = \frac{1}{1 + e^{(\theta_k - p_k)/T}} \]

where \( T \) is called pseudo temperature.
- In equilibrium stochastic networks are more likely to be in a state with low energy.
- Kirkpatrick et al. 1983: Can we compute a global minima?
- Simulated annealing decrease \( T \) gradually.
- Theorem (Geman, Geman 1984)
  A global minima is reached if \( T \) is decreased in infinitesimal small steps.
- Applications Combinatorial optimization problems like the travelling salesman problem or graph bipartitioning problem.
Literature


Propositional Logic

- **Variables** are \( p_1, \ldots, p_n \).
- **Connectives** are \( \neg, \lor, \land \).
- **Atoms** are variables.
- **Literals** are atoms and negated atoms.
- **Clauses** are (generalized) disjunctions of literals.
- **Formulas in clause form** are (generalized) conjunctions of clauses.

**Notation** Sometimes variables are denoted by different letters if there is a bijection between the set of these letters and \( \{ p_1, \ldots, p_n \} \).

**Example**

\[
(o \rightarrow m) \land (s \rightarrow \neg m) \land (c \rightarrow m) \land (c \rightarrow s) \land (v \rightarrow \neg m), \\
(\neg o \lor m) \land (\neg s \lor \neg m) \land (\neg c \lor m) \land (\neg c \lor s) \land (\neg v \lor \neg m), \\
\langle [\neg o, m], [\neg s, \neg m], [\neg c, m], [\neg c, s], [\neg v, \neg m] \rangle.
\]
Interpretations and Models

- **Notation** (all symbols may be indexed)
  - $A$ denotes an atom.
  - $L$ denotes a literal.
  - $F, G$ denote formulas.
  - $C$ denotes a clause.

- **Interpretations** are mappings from $\{p_1, \ldots, p_n\}$ to $\{0, 1\}$.
  - They can be encoded as $\vec{v}$.
  - They are extended to formulas as follows:
    
    $p_i(\vec{v}) = v_i$
    
    $(\neg F)(\vec{v}) = 1 - F(\vec{v})$
    
    $(F \land G)(\vec{v}) = F(\vec{v}) \times G(\vec{v})$
    
    $(F \lor G)(\vec{v}) = F(\vec{v}) + G(\vec{v}) - F(\vec{v}) \times G(\vec{v})$

- $\vec{v}$ is a **model** for $F$ iff $F(\vec{v}) = 1$.

- $F$ is **satisfiable** if it has a model.
Interpretations and Models – Example

Let $F = \langle [\neg p_1, p_2], [p_3, \neg p_2] \rangle$ and $\vec{v} = 1\bar{0}1$, then:

$$F(\vec{v}) = [\neg p_1, p_2](\vec{v}) \times [p_3, \neg p_2](\vec{v})$$
$$= ((\neg p_1)(\vec{v}) + p_2(\vec{v}) - (\neg p_1)(\vec{v}) \times p_2(\vec{v}))$$
$$\times (p_3(\vec{v}) + (\neg p_2)(\vec{v}) - p_3(\vec{v}) \times (\neg p_2)(\vec{v}))$$
$$= ((1 - p_1(\vec{v})) + p_2(\vec{v}) - (1 - p_1(\vec{v})) \times p_2(\vec{v}))$$
$$\times (p_3(\vec{v}) + (1 - p_2(\vec{v})) - p_3(\vec{v}) \times (1 - p_2(\vec{v})))$$
$$= ((1 - 1) + 0 - (1 - 1) \times 0) \times (1 + (1 - 0) - 1 \times (1 - 0))$$
$$= 0 \times 1$$
$$= 0$$

Hence, $\vec{v} = 1\bar{0}1$ is not a model for $F$, but is a model for $[p_3, \neg p_2]$.

Exercise

▷ Is $F$ satisfiable? Prove your claim.
▷ Is $\langle [\neg p], [p, \neg q], [q] \rangle$ satisfiable? Prove your claim.
▷ Find all models of $\langle [\neg o, m], [\neg s, \neg m], [\neg c, m], [\neg c, s], [\neg v, \neg m] \rangle$. 
Propositional Reasoning and Energy Minimization

▶ Pinkas 1991:
Is there a link between propositional logic and symmetric networks?
▶ Let $F = \langle C_1, \ldots, C_m \rangle$ be a propositional formula in clause form.
▶ We define

$$\tau(C) = \begin{cases} 
0 & \text{if } C = [], \\
A & \text{if } C = [A], \\
1 - A & \text{if } C = [\neg A], \\
\tau(C_1) + \tau(C_2) - \tau(C_1)\tau(C_2) & \text{if } C = (C_1 \lor C_2).
\end{cases}$$

$$\tau(F) = \sum_{i=1}^{m} (1 - \tau(C_i))^2.$$  

▶ Example $\tau(\langle [\neg o, m], [\neg s, \neg m], [\neg c, m], [\neg c, s], [\neg v, \neg m] \rangle)$
$= vm - cm - cs + sm - om + 2c + o.$

▶ Exercise Compute $\tau(\langle [\neg p], [p, \neg q], [q] \rangle)$.
Propositional Reasoning and Symmetric Networks

▶ **Theorem** $F(\vec{v}) = 1$ iff $\tau(F)$ has a global minima at $\vec{v}$ and $\tau(F)(\vec{v}) = 0$.

▶ **Compare** $\tau(F) = \begin{align*} vm - cm - cs + sm - om + 2c + o \end{align*}$

with $E(\vec{v}) = - \sum_{k<j} w_{kj} v_j v_k + \sum_k \theta_k v_k$.

![Diagram](image)
Propositional Non-Monotonic Reasoning

- Pinkas 1991a: Can the above mentioned approach be extended to non-monotonic reasoning?
- Consider \( F = \langle (C_1, k_1), \ldots, (C_m, k_m) \rangle \), where \( C_i \) are clauses and \( k_i \in \mathbb{R}^+ \).
- The penalty of \( \vec{v} \) for \( (C, k) \) is \( k \) if \( C(\vec{v}) = 0 \) and 0 otherwise.
- The penalty of \( \vec{v} \) for \( F \) is the sum of the penalties for \( (C_i, k_i) \).
- \( \vec{v} \) is preferred over \( \vec{w} \) wrt \( F \) if the penalty of \( \vec{v} \) for \( F \) is smaller than the penalty of \( \vec{w} \) for \( F \).
- Modify \( \tau \) to become \( \tau(F) = \sum_{i=1}^{m} k_i (1 - \tau(C_i)) \), e.g.,

\[
\tau(\langle ([\neg o, m], 1), ([\neg s, \neg m], 2), ([\neg c, m], 4), ([\neg c, s], 4), ([\neg v, \neg m], 4) \rangle) = 4vm - 4cm - 4cs + 2sm - om + 8c + o.
\]

- The corresponding stochastic network computes most preferred interpretations.
Exercises

► Exercise  Consider

\[ F = \langle ([\neg o, m], 1), ([\neg s, \neg m], 2), ([\neg c, m], 4), ([\neg c, s], 4), ([\neg v, \neg m], 4) \rangle. \]

► Compute the most preferred interpretations of \( F \).

► What happens if we add \((o, 100)\) to \( F \)?

► What happens if we add \((o, 100)\) and \((s, 100)\) to \( F \)?
Literature


Propositional Logic Programs and the Core Method

- The Neural-Symbolic Cycle
- The Very Idea
- Logic Programs
- Propositional Core Method
- Backpropagation
- Knowledge-Based Artificial Neural Networks
- Propositional Core Method using Sigmoidal Units
- Further Extensions
The Neural-Symbolic Cycle

Symbolic System

Connectionist System

writable

embedding

trainable

readable

extraction
The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators (e.g., Apt, vanEmden 1982).

- **Banach Contraction Mapping Theorem** A contraction mapping \( f \) defined on a complete metric space \( (X, d) \) has a unique fixed point. The sequence \( y, f(y), f(f(y)), \ldots \) converges to this fixed point for any \( y \in X \).

  - Fitting 1994: Consider logic programs, whose immediate consequence operator is a contraction.

- **Funahashi 1989**: Every continuous function on the reals can be uniformly approximated by feed-forward connectionist networks.

  - Hölldobler, Kalinke, Störr 1999: Consider logic programs, whose immediate consequence operator is continuous on the reals.
Metrics

- **A metric** on a space $M$ is a mapping $d : M \times M \to \mathbb{R}$ such that
  - $d(x, y) = 0$ iff $x = y$,
  - $d(x, y) = d(y, x)$, and
  - $d(x, y) \leq d(x, z) + d(z, y)$.

- Let $(M, d)$ be a metric space and $S = (s_i \mid s_i \in M)$ a sequence.
  - $S$ **converges** if $(\exists s \in M)(\forall \epsilon > 0)(\exists N)(\forall n \geq N) d(s_n, s) \leq \epsilon$.
  - $S$ is **Cauchy** if $(\forall \epsilon > 0)(\exists N)(\forall n, m \geq N) d(s_n, s_m) \leq \epsilon$.
  - $(M, d)$ is **complete** if every Cauchy sequence converges.

- A mapping $f : M \to M$ is a **contraction** on $(M, d)$ if $(\exists 0 < k < 1)(\forall x, y \in M) d(f(x), f(y)) \leq k \times d(x, y)$. 
Propositional Logic Programs

- A propositional logic program \( \mathcal{P} \) over a propositional language \( \mathcal{L} \)
is a finite set of clauses

\[
A \leftarrow L_1 \land \ldots \land L_n,
\]

where \( A \) is an atom, \( L_i \) are literals and \( n \geq 0 \).
\( \mathcal{P} \) is definite if all \( L_i, 1 \leq i \leq n \), are atoms.

- Let \( \mathcal{V} \) be the set of all propositional variables occurring in \( \mathcal{L} \).
- An interpretation \( I \) is a mapping \( \mathcal{V} \rightarrow \{\top, \bot\} \).
- \( I \) can be represented by the set of atoms which are mapped to \( \top \) under \( I \).
- \( 2^\mathcal{V} \) is the set of all interpretations.
- Immediate consequence operator \( T_\mathcal{P} : 2^\mathcal{V} \rightarrow 2^\mathcal{V} : \)

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

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

► Consider $\mathcal{P} = \{p, q \leftarrow p, r \leftarrow q\}$
  ▶ Draw the lattice of all interpretations of $\mathcal{P}$ wrt the $\subseteq$ ordering.
  ▶ Mark the models of $\mathcal{P}$.
  ▶ Compute $T_{\mathcal{P}}(\emptyset), T_{\mathcal{P}}(T_{\mathcal{P}}(\emptyset)), \ldots$.
  ▶ Mark the supported models of $\mathcal{P}$.

► Let $\mathcal{P}$ be a definite program.
  ▶ Show that if $M_1$ and $M_2$ are models of $\mathcal{P}$ then so is $M_1 \cap M_2$.
  ▶ Let $M$ be the least model of $\mathcal{P}$. Show that $M$ is a supported model.
The Core Method

- Let $\mathcal{L}$ be a logic language.
- Given a logic program $\mathcal{P}$ together with immediate consequence operator $T_\mathcal{P}$.
- Let $\mathcal{I}$ be the set of interpretations for $\mathcal{P}$.
- Find a mapping $\iota: \mathcal{I} \rightarrow \mathbb{R}^n$.
- Construct a feed-forward network computing $f_\mathcal{P}: \mathbb{R}^n \rightarrow \mathbb{R}^n$, called the core, such that the following holds:
  - If $T_\mathcal{P}(I) = J$ then $f_\mathcal{P}(\iota(I)) = \iota(J)$, where $I, J \in \mathcal{I}$.
  - If $f_\mathcal{P}(\vec{s}) = \vec{t}$ then $T_\mathcal{P}(\iota^{-1}(\vec{s})) = \iota^{-1}(\vec{t})$, where $\vec{s}, \vec{t} \in \mathbb{R}^n$.
- 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 or approximates $\iota(I)$.

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.
Let $\mathcal{L}$ be the language of propositional logic over a set $\mathcal{V}$ of variables.

Let $\mathcal{P}$ be a propositional logic program, e.g.,

$$
\mathcal{P} = \{ p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q \}.
$$

$\mathcal{I} = 2^{\mathcal{V}}$ is the set of interpretations for $\mathcal{P}$.

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

$T_\mathcal{P}(\emptyset) = \{ p \}$
$T_\mathcal{P}(\{ p \}) = \{ p, r \}$
$T_\mathcal{P}(\{ p, r \}) = \{ p, r \} = \text{lfp}(T_\mathcal{P})$
Representing Interpretations

- $\mathcal{I} = 2^\mathcal{V}$
- Let $n = |\mathcal{V}|$ and identify $\mathcal{V}$ with $\{1, \ldots, n\}$.
- Define $\iota : \mathcal{I} \rightarrow \mathbb{R}^n$ such that for all $1 \leq j \leq n$ we find:

$$\iota(I)[j] = \begin{cases} 1 & \text{if } j \in I, \\ 0 & \text{if } j \not\in I. \end{cases}$$

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

$$\iota(I)[j] = \begin{cases} 1 & \text{if } j \in I, \\ -1 & \text{if } j \not\in I. \end{cases}$$
Computing the Core

- Consider again $\mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- A translation algorithm translates $\mathcal{P}$ into a core of binary threshold units:

![Diagram of a neural network with input, hidden, and output layers, showing weights and connections.]

- Exercise Specify the core for $\{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6\}$. 

43
Some Results

- **Proposition** 2-layer networks cannot compute $T_{\mathcal{P}}$ for definite $\mathcal{P}$.
- **Theorem** For each program $\mathcal{P}$, there exists a core computing $T_{\mathcal{P}}$.
- **Recall** $\mathcal{P} = \{ p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q \}$.
- **Adding recurrent connections:**
Strongly Determined Programs

- A logic program $\mathcal{P}$ is said to be **strongly determined** if there exists a metric $d$ on the set of all Herbrand interpretations for $\mathcal{P}$ such that $T_\mathcal{P}$ is a contraction wrt $d$.

- **Exercise** Are the following programs strongly determined?
  - $\{p, q \leftarrow p, r \leftarrow q\}$,
  - $\{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6\}$,
  - $\{p \leftarrow \neg p\}$.

- **Corollary** Let $\mathcal{P}$ be a strongly determined program. Then there exists a core with recurrent connections such that the computation with an arbitrary initial input converges and yields the unique fixed point of $T_\mathcal{P}$. 
Time and Space Complexity

- Let \( n \) be the number of clauses and \( m \) be the number of propositional variables occurring in \( \mathcal{P} \).
  - \( 2m + n \) units, \( 2mn \) connections in the core.
  - \( T_P(I) \) is computed in 2 steps.
  - The parallel computational model to compute \( T_P(I) \) is optimal.
  - The recurrent network settles down in \( 3n \) steps in the worst case.

- **Exercise** Give an example of a program with worst case time behavior.
Bipolar Binary Threshold Units

- \( u_k \) is a bipolar binary threshold unit if

\[
\Phi(\vec{i}_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j \\
\Psi(p_k) = v_k = \begin{cases} 
1 & \text{if } p_k \geq \theta_k \\
-1 & \text{otherwise}
\end{cases}
\]

where \( \theta_k \in \mathbb{R} \) is a threshold.

- In the following section on rule extraction bipolar binary threshold units will be used.
Rule Extraction - A Naive Pedagogical Approach

\[ P = \{ g \leftarrow \neg a \land b, \quad h \leftarrow \neg a \land b, \quad g \leftarrow a \land \neg b, \quad h \leftarrow a \land \neg b, \quad h \leftarrow \neg a \land \neg b, \quad h \leftarrow a \land b \} \]

simplified form:

\[ P = \{ g \leftarrow \neg a \land b, \quad g \leftarrow a \land \neg b, \quad h \} \]
A network consisting of a set of input units connected to a single output unit $\alpha$ is called a perceptron (denoted $\mathcal{N}_\alpha$).
Input Patterns

- Let $N_\alpha$ be a perceptron and $U$ the set of input units. $I \subseteq U$ is called input pattern.

- The minimal input wrt $I$ is defined as
  \[ i_{\text{min}}(I) = \sum_{i \in I} w_{ai} - \sum_{u \in U \setminus I} |w_{au}|. \]

- The maximal input wrt $I$ is defined as
  \[ i_{\text{max}}(I) = \sum_{i \in I} w_{ai} + \sum_{u \in U \setminus I} |w_{au}|. \]

\[ i_{\text{min}}(\{c, d\}) = 1.0 + 2.0 - 3.0 - 5.0 = -5.0, \]

\[ i_{\text{max}}(\{c, d\}) = 1.0 + 2.0 + 3.0 + 5.0 = 11.0.\]
Coalitions

Let $\mathcal{N}_\alpha$ be a perceptron with threshold $\theta$ and $I$ be some input pattern for $\mathcal{N}_\alpha$.

- $I$ is called a coalition, if $i_{\text{min}}(I) \geq \theta$.
- A coalition is minimal, if none of its subsets is a coalition.

In other words, a coalition of $\mathcal{N}_\alpha$ activates $\alpha$.

$\{c, d, e, f\}$, $\{c, d, f\}$ and $\{e, f\}$ are coalitions with $\{c, d, f\}$ and $\{e, f\}$ being minimal.
Oppositions

Let $\mathcal{N}_a$ be a perceptron with threshold $\theta$ and $I$ be some input pattern for $\mathcal{N}_a$.

- $I$ is called an opposition, if $\max(I) < \theta$.
- An opposition is minimal, if none of its subsets is an opposition.

In other words, an opposition of $\mathcal{N}_a$ prevents $a$ from becoming active.

\[\neg c, \neg d, \neg e, \neg f\]

\{-f\}, \{-e, \neg d\}, \{-e, \neg c\} and all their supersets are oppositions with 
\{-f\}, \{-e, \neg d\} and \{-e, \neg c\} being minimal.
An Example

<table>
<thead>
<tr>
<th>Minimal Coalitions</th>
<th>Minimal Opp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>c \quad {{a}, {b}}</td>
<td>{{¬a, ¬b}}</td>
</tr>
<tr>
<td>d \quad {{¬a, b}}</td>
<td>{{a}, {¬b}}</td>
</tr>
<tr>
<td>e \quad {{a, ¬b}}</td>
<td>{{¬a}, {b}}</td>
</tr>
<tr>
<td>f \quad {\emptyset}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>g \quad {{e, f}, {c, d, f}}</td>
<td>{{¬f}, {¬e, ¬d}, {¬e, ¬c}}</td>
</tr>
<tr>
<td>h \quad {{¬d}, {f}}</td>
<td>{{d, ¬f}}</td>
</tr>
</tbody>
</table>

\[ g \leftarrow (e \land f) \lor (c \land d \land f) \]
\[ \leftrightarrow (a \land ¬b) \lor ((a \lor b) \land (¬a \land b)) \]
\[ \leftrightarrow (a \land ¬b) \lor (¬a \land b) \]

\[ h \leftarrow ¬d \lor f \]
\[ \leftrightarrow \top \]
COOP: Final Remarks

- For more details see Bader, Hölldobler, Mayer-Eichberger 2007.
- Computations are by need.
- The extraction algorithm is exponential in the worst-case, but shows good average-case behavior in a prototypical implementation.
- Future work:
  - In-depth comparison to other approaches.
  - Extension to bipolar sigmoidal units.
Literature


3-Layer Feed-Forward Networks Revisited

► **Theorem (Funahashi 1989)** Suppose that $\Psi : \mathbb{R} \to \mathbb{R}$ is non-constant, bounded, monotone increasing and continuous. Let $K \subseteq \mathbb{R}^n$ be compact, let $f : K \to \mathbb{R}$ be continuous, and let $\varepsilon > 0$. Then there exists a 3-layer feed-forward network with output function $\Psi$ for the hidden layer and linear output function for the input and output layer whose input-output mapping $\overline{f} : K \to \mathbb{R}$ satisfies

$$\max_{x \in K} | f(x) - \overline{f}(x) | < \varepsilon.$$ 

► Every continuous function $f : K \to \mathbb{R}$ can be uniformly approximated by input-output functions of 3-layer feed-forward networks.

► $u_k$ is a sigmoidal unit if

$$\Phi(i_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j$$

$$\Psi(p_k) = v_k = \frac{1}{1+e^{\beta(\theta_k-p_k)}}$$

where $\theta_k \in \mathbb{R}$ is a threshold (or bias) and $\beta > 0$ a steepness parameter.
Backpropagation


► Training set of input-output pairs \( \{(\vec{i}^l, \vec{o}^l) \mid 1 \leq l \leq n\} \).

► Minimize \( E = \sum_l E^l \) where \( E^l = \frac{1}{2} \sum_k (o_k^l - v_k^l)^2 \).

► Gradient descent algorithm to learn appropriate weights.

► Backpropagation

▷ Initialize weights arbitrarily.

▷ Do until all input-output patterns are correctly classified.

1. Present input pattern \( \vec{i}^l \) at time \( t \).
2. Compute output pattern \( \vec{v}^l \) at time \( t + 2 \).
3. Change weights according to \( \Delta w_{ij}^l = \eta \delta_i^l v_j^l \), where

\[
\delta_i^l = \left\{ \begin{array}{ll}
\Psi_i'(p_i^l) \times (o_i^l - v_i^l) & \text{if } i \text{ is output unit}, \\
\Psi_i'(p_i^l) \times \sum_k \delta_k^l w_{ki} & \text{if } i \text{ is hidden unit}, 
\end{array} \right.
\]

\( \eta > 0 \) is called learning rate.
Output Functions Revisited

▸ Remember sigmoidal function (with \( \beta = 1 \)):

\[
v_i = \frac{1}{1 + e^{-(\sum_j w_{ij} v_j + \theta_i)}}
\]

▸ We find

\[
\frac{dv_i}{d(\sum_j w_{ij} v_j + \theta_i)} = v_i(1 - v_i).
\]

▸ Hence

\[
\delta_i = \begin{cases} 
  v_i^l(1 - v_i^l)(o_i^l - v_i^l) & \text{if } u_i \text{ is an output unit,} \\
  v_i^l(1 - v_i^l)\sum_k \delta_k^l w_{ki} & \text{if } u_i \text{ is a hidden unit.}
\end{cases}
\]

▸ When is a unit active?

▸ Let \( \alpha \in [0, 0.5] \).

▸ Units are active if its value \( v \in [\alpha, 1] \).

▸ Units are passive if its value \( v \in [0, \alpha] \).
Properties

- Learning rate $\eta$:
  - If $\eta$ is large, then system learns rapidly but may oscillate.
  - If $\eta$ is small, then system learns slowly but will not oscillate.
  - In the ideal case $\eta$ should be adapted during learning:
    \[
    \Delta w_{ij}(t + 1) = \eta \delta_i(t) v_j(t) + \alpha \Delta w_{ij}(t)
    \]
    where $\alpha$ is a constant and $\alpha \Delta w_{ij}(t)$ is called momentum term.

- Almost all functions can be learned.
- Learning is NP-hard.

Let $\mathcal{V}$ be a set of propositional variables and $\mathcal{P}$ be a propositional logic program wrt $\mathcal{V}$.

A level mapping for $\mathcal{P}$ is a function $l : \mathcal{V} \to \mathbb{N}$.

We define $l(\neg A) = l(A)$.

$\mathcal{P}$ is hierarchical if for all clauses $A \leftarrow L_1 \land \ldots \land L_n \in \mathcal{P}$ we find $l(A) > l(L_i)$ for all $1 \leq i \leq n$.

Exercise Are the following programs hierarchical?

- $\{p, q \leftarrow p, r \leftarrow q \}$,
- $\{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6 \}$,
- $\{p \leftarrow \neg p \}$. 

Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a \land \neg h\}. \]
Knowledge Based Artificial Neural Networks – Learning

- Given hierarchical 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.
Knowledge Based Artificial Neural Networks – A Problem

► “Works if rules have few conditions and there are few rules with the same head.”

► \( p_q = p_r = 9\omega \) and \( v_q = v_r = \frac{1}{1+e^{\beta(9.5\omega-9\omega)}} \approx 0.46 \) with \( \beta = 1 \).

► \( p_s = 0.92\omega \) and \( v_s = \frac{1}{1+e^{\beta(0.5\omega-0.92\omega)}} \approx 0.6 \) with \( \beta = 1 \).
Propositional Core Method using Bipolar Sigmoidal Units

- d’Avila Garcez, Zaverucha, Carvalho 1997:
  Can we combine the ideas in Hölldobler, Kalinke 1994 and Towell, Shavlik 1994 while avoiding the above mentioned problem?

- Consider propositional logic language.

- Let \( I \) be an interpretation and \( a \in [0, 1] \).

\[
\iota(I)[j] = \begin{cases} 
  v \in [a, 1] & \text{if } j \in I, \\
  w \in [-1, -a] & \text{if } j \not\in I.
\end{cases}
\]

- Replace threshold and sigmoidal units by bipolar sigmoidal ones, i.e., units with

\[
\Phi(\vec{i}_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j,
\]

\[
\Psi(p_k) = v_k = \frac{2}{1 + e^{\beta(\theta_k - p_k)}} - 1,
\]

where \( \theta_k \in \mathbb{R} \) is a threshold (or bias) and \( \beta > 0 \) a steepness parameter.
The Task

▶ How should $a$, $\omega$ and $\theta_i$ be selected such that:

▶ $v_i \in [a, 1]$ or $v_i \in [-1, -a]$ and

▶ the core computes the immediate consequence operator?
Hidden Layer Units

- Consider $A \leftarrow L_1 \land \ldots \land L_n$.
- Let $u$ be the hidden layer unit for this rule.
  - Suppose $I \models L_1 \land \ldots \land L_n$.
    - $u$ receives input $\geq \omega a$ from unit representing $L_i$.
    - $p_u \geq n\omega a = p_u^+$.
  - Suppose $I \not\models L_1 \land \ldots \land L_n$.
    - $u$ receives input $\leq -\omega a$ from at least one unit representing $L_i$.
    - $p_u \leq (n - 1)\omega 1 - \omega a = p_u^-$.
  - $\theta_u = \frac{n\omega a + (n-1)\omega - \omega a}{2} = (na + n - 1 - a)\frac{\omega}{2} = (n - 1)(a + 1)\frac{\omega}{2}$.
Output Layer Units

- Let $\mu$ be the number of clause with head $A$.
- Consider $A \leftarrow L_1 \land \ldots \land L_n$.
- Suppose $I \models L_1 \land \ldots \land L_n$.
  - $\triangleright p_A \geq \omega a + (\mu - 1)\omega(-1) = \omega a - (\mu - 1)\omega = p^+_A$.
- Suppose for all rules of the form $A \leftarrow L_1 \land \ldots \land L_n$ we find $I \nvdash L_1 \land \ldots \land L_n$.
  - $\triangleright p_A \leq -\mu \omega a = p^-_A$.
- $\theta_A = \frac{\omega a - (\mu - 1)\omega - \mu \omega a}{2} = (a - \mu + 1 - \mu a)\frac{\omega}{2} = (1 - \mu)(a + 1)\frac{\omega}{2}$. 


Computing a Value for $a$

$\Delta p^+_u > p^-_u$:
- $n\omega a > (n - 1)\omega - \omega a$.
- $n\omega a + \omega a > (n - 1)\omega$.
- $a(n + 1)\omega > (n - 1)\omega$.
- $a > \frac{n-1}{n+1}$.

$\Delta p^+_A > p^-_A$:
- $\omega a - (\mu - 1)\omega > -\mu a\omega$.
- $\omega a + \mu a\omega > (\mu - 1)\omega$.
- $a(1 + \mu)\omega > (\mu - 1)\omega$.
- $a > \frac{\mu-1}{\mu+1}$.

Consider all rules $\Rightarrow$ minimum value for $a$. 

Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq a$.
- $\frac{2}{1+e^{\beta(\theta-p)}} \geq 1 + a$.
- $\frac{2}{1+a} \geq 1 + e^{\beta(\theta-p)}$.
- $\frac{2}{1+a} - 1 = \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta-p)}$.
- $\ln(\frac{1-a}{1+a}) \geq \beta(\theta - p)$.
- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq \theta - p$.

Consider a hidden layer unit:

- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq (n - 1)(a + 1)\frac{\omega}{2} - n\omega a = \frac{na+n-a-1-2na}{2} \omega = \frac{n-1-a(n+1)}{2} \omega$.
- $\omega \geq \frac{2}{(n-1-a(n+1))\beta} \ln(\frac{1-a}{1+a})$ because $a \geq \frac{n-1}{n+1}$.

Consider all hidden and output layer units as well as the case that $\Psi(p) \leq -a$:

minimum value for $\omega$. 

\begin{align*}
\Psi(p) &= \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq a, \\
\frac{2}{1+e^{\beta(\theta-p)}} &\geq 1 + a, \\
\frac{2}{1+a} &\geq 1 + e^{\beta(\theta-p)}, \\
\frac{2}{1+a} - 1 &= \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta-p)}, \\
\ln(\frac{1-a}{1+a}) &\geq \beta(\theta - p), \\
\frac{1}{\beta} \ln(\frac{1-a}{1+a}) &\geq \theta - p.\
\end{align*}
Exercises

▶ Show that hierarchical programs are strongly determined.
▶ Consider \( \mathcal{P} = \{r \leftarrow p \land \neg q, r \leftarrow \neg p \land q, p \leftarrow s \land t\} \).
  ▶ Compute values for \( a, \omega \) and \( \theta_i \).
  ▶ Specify the core for \( \mathcal{P} \).
  ▶ How can the approach be extended to handle facts like \( s \) and \( t \)?
▶ Consider now \( \mathcal{P}' = \mathcal{P} \cup \{s, t\} \), where \( \mathcal{P} \) is as before.
  ▶ Show that \( \mathcal{P}' \) is strongly determined.
  ▶ Show that the recurrent network computes the least model of \( \mathcal{P} \cup \{s, t\} \).
Results

- Relation to logic programs is preserved.
- The core is trainable by backpropagation.
- Many interesting applications, e.g.:
  - DNA sequence analysis.
  - Power system fault diagnosis.
- Empirical evaluation:
  system performs better than well-known machine learning systems.
- See d’Avila Garcez, Broda, Gabbay 2002 for details.
Literature


Further Extensions

- Many-valued logic programs
- Modal logic programs
- Answer set programming
- Metalevel priorities
- Rule extraction
Propositional Core Method – Three-Valued Logic Programs

- **Kalinke 1994**: Consider truth values $\top$, $\bot$, $u$.
- Interpretations are pairs $I = \langle I^+, I^- \rangle$.
- Immediate consequence operator $\Phi_P(I) = \langle J^+, J^- \rangle$, where
  \[
  J^+ = \{ A \mid A \leftarrow L_1 \land \ldots \land L_m \in \mathcal{P} \text{ and } I(L_1 \land \ldots \land L_m) = \top \},
  \]
  \[
  J^- = \{ A \mid \text{for all } A \leftarrow L_1 \land \ldots \land L_m \in \mathcal{P} : I(L_1 \land \ldots \land L_m) = \bot \}.\]
- Let $n = |\mathcal{N}|$ and identify $\mathcal{N}$ with $\{1, \ldots, n\}$.
- Define $\iota : \mathcal{I} \rightarrow \mathbb{R}^{2n}$ as follows:
  \[
  \iota(I)[2j - 1] = \begin{cases} 
  1 & \text{if } j \in I^+ \\
  0 & \text{if } j \notin I^+ 
  \end{cases} \quad \text{and} \quad \iota(I)[2j] = \begin{cases} 
  1 & \text{if } j \in I^- \\
  0 & \text{if } j \notin I^- 
  \end{cases}
  \]
Propositional Core Method – Multi-Valued Logic Programs

▶ For each program \( \mathcal{P} \), there exists a core computing \( \Phi_{\mathcal{P}} \), e.g.,

\[
\mathcal{P} = \{ C \leftarrow A \land \neg B, D \leftarrow C \land E, D \leftarrow \neg C \}. 
\]

▶ Seda, Lane 2004: Extension to finitely determined sets of truth values.
Propositional Core Method – Modal Logic Programs

▶ Let $L$ be a propositional logic language plus
  ▶ the modalities $\Box$ and $\Diamond$, and
  ▶ a finite set of labels $w_1, \ldots, w_k$ denoting worlds.
▶ Let $B$ be an atom, then $\Box B$ and $\Diamond B$ are modal atoms.
▶ A modal definite logic program $P$ is a set of clauses of the form

$$w_i : A \leftarrow A_1 \land \ldots \land A_m$$

together with a finite set of relations $w_i \triangleright w_j$, where
$w_i$, $1 \leq i, j \leq k$, are labels and $A, A_1, \ldots, A_m$ are atoms or modal atoms.
▶ $P = \bigcup_{i=1}^{k} P_i$, where $P_i$ consists of all clauses labelled with $w_i$. 
Modal Logic Programs – Semantics

Example: \( \mathcal{P} = \{ w_1 : A, w_1 : \Diamond C \leftarrow A \} \cup \{ w_2 : B \} \cup \{ w_3 : B \} \cup \{ w_4 : B \} \cup \{ w_1 \uparrow w_2, w_1 \uparrow w_3, w_1 \uparrow w_4, w_2 \uparrow w_4, \} \)

Kripke semantics:

\[
f_C(w_1) = w_4
\]
Modal Immediate Consequence Operator

- Interpretations are tuples $I = \langle I_1, \ldots, I_k \rangle$
- Immediate consequence operator $MT_{\mathcal{P}}(I) = \langle J_1, \ldots, J_k \rangle$, where

\[
J_i = \begin{cases} 
\{ A \mid \text{there exists } A \leftarrow A_1 \land \ldots \land A_m \in \mathcal{P}_i \\
\text{such that } \{ A_1, \ldots, A_m \} \subseteq I_i \} \\
\cup \{ \Diamond A \mid \text{there exists } w_i \triangleright w_j \in \mathcal{P} \text{ and } A \in I_j \} \\
\cup \{ \Box A \mid \text{for all } w_i \triangleright w_j \in \mathcal{P} \text{ we find } A \in I_j \} \\
\cup \{ A \mid \text{there exists } w_j \triangleright w_i \in \mathcal{P} \text{ and } \Box A \in I_j \} \\
\cup \{ A \mid \text{there exists } w_j \triangleright w_i \in \mathcal{P}, \Diamond A \in I_j \text{ and } f_A(w_j) = w_i \} 
\end{cases}
\]
Modal Logic Programs – The Translation Algorithm

- Let \( n = |\mathcal{V}| \) and identify \( \mathcal{V} \) with \( \{1, \ldots, n\} \).
- Let \( a \in [0, 1] \).
- Define \( \iota : \mathcal{I} \rightarrow \mathbb{R}^{3n} \) as follows:

\[
\iota(I)[3j - 2] = \begin{cases} 
    v \in [a, 1] & \text{if } j \in I_j \\
    w \in [-1, -a] & \text{if } j \not\in I_j
\end{cases}
\]

\[
\iota(I)[3j - 1] = \begin{cases} 
    v \in [a, 1] & \text{if } \Box j \in I_j \\
    w \in [-1, -a] & \text{if } \Box j \not\in I_j
\end{cases}
\]

\[
\iota(I)[3j] = \begin{cases} 
    v \in [a, 1] & \text{if } \Diamond j \in I_j \\
    w \in [-1, -a] & \text{if } \Diamond j \not\in I_j
\end{cases}
\]

- Translation algorithm such that
  - for each world the “local” part of \( MT_P \) is computed by a core,
  - the cores are turned into recurrent networks, and
  - the cores are connected with respect to the given set of relations.
The Example Network

$w_1$

$w_2$

$w_3$

$w_4$
Literature

