Now we need only translate the objective function. To this end, we use $P^{l-1}_{(s')\,i} P^{l-1}_{(s)\,i} = \delta_{(s')\,(s)} P^{l-1}_{(s)\,i}$ and $P^{l}_{(s'\sigma)\,j} P^{l}_{(s\sigma)\,j} = \delta_{(s')\,(s)} P^{l}_{(s\sigma)\,j}$ to calculate

$$\begin{aligned}
\sum_{\alpha\beta} \sum_{ij} \sum_{(s\sigma)} \mathrm{INA}_{\alpha\sigma\beta} A^{l\beta}_{(s\sigma)} A^{l-1\alpha}_{(s)} P^{l}_{(s\sigma)\,j} P^{l-1}_{(s)\,i} H^{(\alpha\sigma\beta)}(x_i, x_j) &= \sum_{\alpha\beta} \sum_{ij} \sum_{(s\sigma)} \mathrm{INA}_{\alpha\sigma\beta} A^{l\beta}_{(s\sigma)} A^{l-1\alpha}_{(s)} P^{l}_{(s\sigma)\,j} P^{l-1}_{(s)\,i} H^{(\alpha\sigma\beta)}(x_i, x_j) \\
&= \sum_{\alpha\sigma\beta} \sum_{ij} \sum_{(ss's'')} \mathrm{INA}_{\alpha\sigma\beta} A^{l\beta}_{(s'\sigma)} A^{l-1\alpha}_{(s'')} \\
&\quad \times P^{l}_{(s'\sigma)\,j} P^{l-1}_{(s'')\,i} P^{l}_{(s\sigma)\,j} P^{l-1}_{(s)\,i} H \\
&= \sum_{\alpha\sigma\beta} \sum_{ij} \mathrm{INA}_{\alpha\sigma\beta} M^{l}_{\beta j} M^{l-1}_{\alpha i} H^{(\alpha\sigma\beta)}(x_i, x_j) \sum_{(s)} P^{l}_{(s\sigma)\,j} P^{l-1}_{(s)\,i} \\
&= \sum_{\alpha\sigma\beta} \sum_{ij} \mathrm{INA}_{\alpha\sigma\beta} \mathrm{ina}^{l}_{i\sigma j} M^{l}_{\beta j} M^{l-1}_{\alpha i} H^{(\alpha\sigma\beta)}(x_i, x_j)
\end{aligned}$$

(where we have used the fact that *H* depends on σ but not (s)) which shows the desired equivalence of objective functions.

The rectangle constraint on ina, INA and M is derived in another calculation:

$$\begin{aligned}
\textstyle\sum_j \mathrm{ina}^l_{i\sigma j} M^l_{\beta j} &= \textstyle\sum_{(s)} \sum_{(s')} \sum_j A^{l\beta}_{(s')} P^l_{(s')j} P^{l-1}_{(s)i} P^l_{(s\sigma)j} \\
&= \textstyle\sum_{(s)} \sum_{(s')} A^{l\beta}_{(s')} P^{l-1}_{(s)i} \sum_j P^l_{(s')j} P^l_{(s\sigma)j} \\
&= \textstyle\sum_{(s)} \sum_{(s')} A^{l\beta}_{(s')} P^{l-1}_{(s)i} \delta_{(s')(s\sigma)} \hat{A}^l_{(s')} \\
&= \textstyle\sum_{(s')} A^{l\beta}_{(s'\sigma)} P^{l-1}_{(s')i} \\
&\le \textstyle\sum_\alpha \mathrm{INA}_{\alpha\sigma\beta} \sum_{(s')} A^{l-1\alpha}_{(s')} P^{l-1}_{(s')i} \\
&= \textstyle\sum_\alpha \mathrm{INA}_{\alpha\sigma\beta} M^{l-1}_{\alpha i}
\end{aligned}$$

So we have shown $C_{A,P} \wedge \Delta_F \Rightarrow C_M$. Next we show $C_{A,P} \wedge \Delta_F \Rightarrow \Delta_B$. For this we must show $A^{l\beta}_{(s)} = \sum_j M^l_{\beta j} P^l_{(s)j}$ and $P^l_{(s\sigma)j} = \sum_i \mathrm{ina}^l_{i\sigma j} P^{l-1}_{(s)i}$. Substituting from Δ_F, $\sum_j M^l_{\beta j} P^l_{(s)j} = \sum_{(s')} A^{l\beta}_{(s')} \sum_j P^l_{(s')j} P^l_{(s)j} = \sum_{(s')} A^{l\beta}_{(s')} \delta_{(s')(s)} \hat{A}^l_{(s)} = A^{l\beta}_{(s)}$. Likewise, $\sum_i \mathrm{ina}^l_{i\sigma j} P^{l-1}_{(s)i} = \sum_{(s')} P^l_{(s'\sigma)j} \sum_i P^{l-1}_{(s')i} P^{l-1}_{(s)i} = \sum_{(s')} P^l_{(s'\sigma)j} \delta_{(s')(s)} \hat{A}^{l-1}_{(s)} = P^l_{(s\sigma)j}$, as desired. In addition, the relation between A and $\hat{A}$ variables is directly present in $C_{A,P}$ and the expression for $P^0_{(\,)i}$ follows from Δ_F: $M^0_{\alpha i} = A^{0\alpha}_{(\,)} P^0_{(\,)i} = \delta_{\alpha \mathrm{root}} P^0_{(\,)i}$. So $C_{A,P} \wedge \Delta_F \Rightarrow \Delta_B$.

Now we need to establish the backwards direction, $C_M \wedge \Delta_B \Rightarrow C_{A,P} \wedge \Delta_F$, starting with $C_M \wedge \Delta_B \Rightarrow C_{A,P}$. That alloc is a 0/1-valued variable is direct. For the A and $\hat{A}$ variables, their expressions in Δ_B are non-negative integers and we can bound them by one at the same time that we establish the summation and 0/1 constraints on P, using induction on l. For the base case, $A^{0\alpha}_{(\,)} \le \hat{A}^0_{(\,)} = \sum_{\alpha i} M^0_{\alpha i} P^0_{(\,)i} = \sum_{\alpha i} M^0_{\alpha i} M^0_{\mathrm{root}\,i} =$ (since $\sum_\alpha M^0_{\alpha i} \le 1$) $\sum_\alpha \delta_{\alpha \mathrm{root}} \sum_i M^0_{\mathrm{root}\,i} = \sum_\alpha (\delta_{\alpha\mathrm{root}})^2 = 1$, as desired. Also, $\sum_{(\,)} P^0_{(\,)i} = P^0_{(\,)i} = M^0_{\mathrm{root}\,i} \le \sum_\alpha M^0_{\alpha i} = \mathrm{alloc}^0_i$; on the other hand $\sum_\alpha M^0_{\alpha i} = M^0_{\mathrm{root}\,i} + \sum_{\alpha \ne \mathrm{root}} M^0_{\alpha i} \le M^0_{\mathrm{root}\,i} + \sum_{\alpha \ne \mathrm{root}} \left(\sum_i M^0_{\alpha i} \right) = M^0_{\mathrm{root}\,i}$, so $\sum_{(\,)} P^0_{(\,)i} = \mathrm{alloc}^0_i \in \{0, 1\}$. From this fact we can also deduce $\hat{A}^0_{(\,)} = \sum_{\alpha j} M^0_{\alpha i} P^0_{(\,)j} = \sum_j \mathrm{alloc}^l_j P^0_{(\,)j} = \sum_j P^0_{(\,)j}$. The induction step is: for A, $A^{l\beta}_{(s\sigma)} \le \hat{A}^l_{(s\sigma)} = \sum_j \left(\sum_\beta M^l_{\beta j} \right) P^l_{(s\sigma)j} = \sum_j \mathrm{alloc}^l_j P^l_{(s\sigma)j} \le \sum_j P^l_{(s\sigma)j} = \sum_{ij} \mathrm{ina}^l_{i\sigma j} P^{l-1}_{(s)i} \le \sum_i \mathrm{alloc}^{l-1}_i P^{l-1}_{(s)i}$ which is, by induction, $= \hat{A}^{l-1}_{(s)}$. For P, $\sum_{(s\sigma)} P^l_{(s\sigma)j} = \sum_{i\sigma} \mathrm{ina}^l_{i\sigma j} \left(\sum_{(s)} P^{l-1}_{(s)i} \right) = \sum_{i\sigma} \mathrm{ina}^l_{i\sigma j} \mathrm{alloc}^{l-1}_i = \sum_{i\sigma} \mathrm{ina}^l_{i\sigma j} = \mathrm{alloc}^l_j$. From this equality, we can also deduce $\hat{A}^l_{(s)} = \sum_{\beta j} M^l_{\beta j} P^l_{(s)j} = \sum_j \mathrm{alloc}^l_j P^l_{(s)j} = \sum_j P^l_{(s)j}$. Finally $P^l_{(s)j} \le \sum_j P^l_{(s)j} = \hat{A}^l_{(s)} \le 1$. This proves all the relevant 0/1-valuedness constraints, and the summation constraints on P.

The value of $A^{0\alpha}_{(\,)}$ may be computed as with $\hat{A}^0_{(\,)}$: $A^{0\alpha}_{(\,)} = \sum_i M^0_{\alpha i} P^0_{(\,)i} = \sum_i M^0_{\alpha i} M^0_{\mathrm{root}\,i} = \delta_{\alpha\mathrm{root}} \sum_i M^0_{\mathrm{root}\,i} = (\delta_{\alpha\mathrm{root}})^2 = \delta_{\alpha\mathrm{root}}$. That $\hat{A}^l_{(s\sigma)} \le \hat{A}^{l-1}_{(s)}$ follows from what has already been proven: $\hat{A}^l_{(s\sigma)} = \sum_j P^l_{(s\sigma)j} = \sum_i \left(\sum_j \mathrm{ina}^l_{i\sigma j} \right) P^{l-1}_{(s)i} \le \sum_i \mathrm{alloc}^{l-1}_i P^{l-1}_{(s)i} = \sum_i P^{l-1}_{(s)i} = \hat{A}^{l-1}_{(s)}$.

The last component of $C_{A,P}$ is the INA inequality. It is computed as follows: $A^{l\beta}_{(s\sigma)} = \sum_j M^l_{\beta j} P^l_{(s\sigma)j} = \sum_i \sum_j \mathrm{ina}^l_{i\sigma j} M^l_{\beta j} P^{l-1}_{(s)i} \le \sum_{\alpha i} \mathrm{INA}_{\alpha\sigma\beta} M^{l-1}_{\alpha i} P^{l-1}_{(s)i} = \sum_\alpha \mathrm{INA}_{\alpha\sigma\beta} \sum_i M^{l-1}_{\alpha i} P^{l-1}_{(s)i} = \sum_\alpha \mathrm{INA}_{\alpha\sigma\beta} A^{l-1\alpha}_{(s)}$. So we have proven $C_M \wedge \Delta_B \Rightarrow C_{A,P}$.

Next we prove $C_M \wedge \Delta_B \Rightarrow \Delta_F$. The expression for ina can be verified simply: $\sum_{(s)} P^l_{(s\sigma)j} P^{l-1}_{(s)i} = \sum_{i'(s)} \mathrm{ina}^l_{i'\sigma j} P^{l-1}_{(s)i'} P^{l-1}_{(s)i} = \sum_{i'} \mathrm{ina}^l_{i'\sigma j} \mathrm{alloc}^{l-1}_i \delta_{ii'} = \mathrm{ina}^l_{i\sigma j} \mathrm{alloc}^{l-1}_i = \mathrm{ina}^l_{i\sigma j}$. The expression for M is: $\sum_{(s\sigma)} A^{l\beta}_{(s\sigma)} P^l_{(s\sigma)j} = \sum_{(s\sigma)j'} M^l_{\beta j'} P^l_{(s\sigma)j'} P^l_{(s\sigma)j} = \sum_{(s\sigma)j'} M^l_{\beta j'} \delta_{j'j} P^l_{(s)j} = \sum_{(s\sigma)} M^l_{\beta j} P^l_{(s\sigma)j} = M^l_{\beta j} \sum_{(s\sigma)} P^l_{(s)j} =$ (by $C_{A,P}$ which has already been proven from $C_M \wedge \Delta_B$) $M^l_{\beta j} \mathrm{alloc}^l_j = M^l_{\beta j}$. The other components of Δ_F are trivial. So we have proven $C_M \wedge \Delta_B \Rightarrow \Delta_F$.

Finally we must translate the objective. We first show $A^{l\beta}_{(s)}A^{l-1\alpha}_{(s)} = \mathrm{INA}_{\alpha s_l \beta}A^{l\beta}_{s}A^{l-1\alpha}_{(s)}$. Since $\sum_\beta A^{l\beta}_{(s)} = \hat{A}^l_{(s)} \in \{0,1\}$, we know that $A^{l\beta}_{(s)}A^{l\beta'}_{(s)} = \delta_{\beta\beta'}A^{l\beta}_{(s)}$, and likewise $A^{l-1\alpha}_{(s)}A^{l-1\alpha'}_{(s)} = \delta_{\alpha\alpha'}A^{l-1\alpha}_{(s)}$. Therefore

$$\begin{aligned}
\mathrm{INA}_{\alpha s_l \beta}A^{l\beta}_{(s)}A^{l-1\alpha}_{(s)} \le A^{l\beta}_{(s)}A^{l-1\alpha}_{(s)} &= A^{l\beta}_{(s)}A^{l\beta}_{(s)}A^{l-1\alpha}_{(s)} \\
&\le A^{l\beta}_{(s)}\left(\sum_{\alpha'}\mathrm{INA}_{\alpha' s_l \beta}A^{l-1\alpha'}_{(s)}\right)A^{l-1\alpha}_{(s)} \\
&= A^{l\beta}_{(s)}\sum_{\alpha'}\mathrm{INA}_{\alpha' s_l \beta}A^{l-1\alpha'}_{(s)}A^{l-1\alpha}_{(s)} \\
&= A^{l\beta}_{(s)}\sum_{\alpha'}\mathrm{INA}_{\alpha' s_l \beta}\delta_{\alpha\alpha'}A^{l-1\alpha}_{(s)} \\
&= \mathrm{INA}_{\alpha s_l \beta}A^{l\beta}_{s}A^{l-1\alpha}_{(s)}
\end{aligned}$$

as desired. Using this fact, and substituting $x^l_{(s)} = \sum_j P^l_{(s)j}x^l_j$, we can calculate

$$\begin{aligned}
A^{l\beta}_{s}A^{l-1\alpha}_{(s)}\left[H^{(\alpha s_l\beta)}\left(x^l_{(s)}, x^{l-1}_{(s)}; u^{(\alpha s_l\beta)}\right) - \mu^{(s_l\beta)}\right] &= \mathrm{INA}_{\alpha s_l \beta}A^{l\beta}_{s}A^{l-1\alpha}_{(s)}\left[H^{(\alpha s_l\beta)}\left(\sum_j P^l_{(s)j}x^l_j, \sum_i P^{l-1}_{(s)i}x^{l-1}_i; u^{(\alpha s_l\beta)}\right) - \mu^{(s_l\beta)}\right] \\
&= \sum_{ij}\mathrm{INA}_{\alpha s_l \beta}A^{l\beta}_{s}A^{l-1\alpha}_{(s)}P^l_{(s)j}P^{l-1}_{(s)i}\left[H^{(\alpha s_l\beta)}\left(x^l_j, x^{l-1}_i; u^{(\alpha s_l\beta)}\right) - \mu^{(s_l\beta)}\right]
\end{aligned}$$

in which the last step follows from the winner-take-all constraints $\sum_j P^l_{(s)j} = \hat{A}^l_{(s)}$ and $\sum_i P^{l-1}_{(s)i} = \hat{A}^{l-1}_{(s)}$ for 0/1 variables, in which $\hat{A}^l_{(s)} = 1$ and $\hat{A}^{l-1}_{(s)} = 1$ for any nonzero instance of the above expressions. Substituting this result into the objective of equation (5) yields the objective of equation (11), as desired. Thus we have proven the validity of the first change of variables.

Now we show the validity of the change of variables (13) from $C_{A,P}$ (equation (12)) to C_M (equation (15)). The constraints on *x*, alloc and INA are unchanged. First we show $C_{A,P} \wedge \Delta_F \Rightarrow C_M$. To show that *M* and ina are 0/1-valued variables, we bound their expressions in Δ_F: $M^l_{\beta j} = \sum_{(s)} A^{l\beta}_{(s)}P^l_{(s)j} \le \sum_{(s)} P^l_{(s)j} = \mathrm{alloc}^l_j \le 1$, and $\mathrm{ina}^l_{is_l j} = \sum_{s_1\ldots s_{l-1}} P^{l-1}_{s_1\ldots s_{l-1},i}P^l_{s_1\ldots s_l,j} \le \sum_{s_1\ldots s_{l-1}} P^{l-1}_{s_1\ldots s_{l-1},i} = \mathrm{alloc}^{l-1}_i \le 1$. The level-0 boundary condition on *M* is: $\sum_i M^l_{\alpha j} = \sum_{i,(s)} A^{0\alpha}_{(\,)}P^0_{(\,)i} = \delta_{\alpha\mathrm{root}}\sum_i P^0_{(\,)i} = \delta_{\alpha\mathrm{root}}\hat{A}^0 = \delta_{\alpha\mathrm{roo}}$, since $\hat{A}^0 = 1$.

For the allocation constraint on *M*, we calculate that $\sum_\beta M^l_{\beta j} = \sum_{(s)}\left(\sum_\beta A^{l\beta}_{(s)}\right)P^l_{(s)j} = \sum_{(s)} P^l_{(s)j} = \mathrm{alloc}^l_j$, where we have used $P^l_{(s)j} \le \sum_j P^l_{(s)j} = \hat{A}^l_{(s)}$. Likewise for the allocation constraints on ina, $\sum_{i\sigma}\mathrm{ina}^l_{i\sigma j} = \sum_{(s)}\left(\sum_i P^{l-1}_{(s)i}\right)P^l_{(s)j}$ $= \sum_{(s)}\hat{A}^{l-1}_{(s)}P^l_{(s)j} = \sum_{(s)} P^l_{(s)j} = \mathrm{alloc}^l_j$, where we have used $\hat{A}^{l-1}_{(s)}P^l_{(s)j} \le P^l_{(s)j} = P^l_{(s)j}P^l_{(s)j} \le P^l_{(s)j}\sum_j P^l_{(s)j} =$ $= P^l_{(s)j}\hat{A}^l_{(s)} \le P^l_{(s)j}\hat{A}^{l-1}_{(s)}$. On the other hand, $\sum_j \mathrm{ina}^l_{i\sigma j} = \sum_{(s)}\left(\sum_j P^l_{(s\sigma)j}\right)P^{l-1}_{(s)i} = \sum_{(s)}\hat{A}^l_{(s\sigma)}P^{l-1}_{(s)i} \le \sum_{(s)}\hat{A}^{l-1}_{(s)}P^{l-1}_{(s)j}$ $= \sum_{(s)} P^{l-1}_{(s)j} = \mathrm{alloc}^{l-1}_i$, where we have used $\hat{A}^{l-1}_{(s)}P^{l-1}_{(s)j} \le P^{l-1}_{(s)j} = P^{l-1}_{(s)j}P^{l-1}_{(s)j} \le P^{l-1}_{(s)j}\sum_j P^{l-1}_{(s)j} = P^{l-1}_{(s)j}\hat{A}^{l-1}_{(s)}$.

$\omega_{(s)}^{l-1} = \omega_{(s)}^{l-1}\omega_{(s)}^{l-1} \le \omega_{(s)}^{l-1}\prod_{k=1}^{l-2}\omega_{(s)}^{k} = \prod_{k=1}^{l-1}\omega_{(s)}^{k}$, which establishes $\sum_\beta C_{(s)}^{l\beta} \le \prod_{k=1}^{l-1}\omega_{(s)}^{k}$ for all *l*. As a corollary, we see that $\prod_{k=1}^{l}\omega_{(s)}^{k} = \omega_{(s)}^{l}$, and therefore the constraints on *x* and *P* in C_C are a direct translation of those in C_A.

The last constraint in C_C is $C_{(s)}^{l\beta} \le \sum_\alpha \mathrm{INA}_{\alpha s_l \beta} C_{(s)}^{l-1\alpha}$, which we prove using Δ_B and nonnegativity of $\mathrm{INA}_{\alpha s_l \beta}$ and $\tilde{C}_{(s)}^{l\beta}$: $\sum_\alpha \mathrm{INA}_{\alpha s_l \beta} C_{(s)}^{l-1\alpha} - C_{(s)}^{l\beta} = \sum_\alpha \mathrm{INA}_{\alpha s_l \beta}\left(A_{(s)}^{l-1\alpha} + \tilde{C}_{(s)}^{l-1\alpha}\right) - A_{(s)}^{l\beta} - \tilde{C}_{(s)}^{l\beta} \ge \sum_\alpha \mathrm{INA}_{\alpha s_l \beta} A_s^{l-1\alpha} - A_{(s)}^{l\beta} - \tilde{C}_{(s)}^{l\beta}$, which is ≥ 0 by C_A.

Next we show that $C_A \wedge \Delta_B \Rightarrow \Delta_F$: The conditions on A^0 follow directly from C_A. $\hat{A}_{(s)}^{l} = \omega_{(s)}^{l}$ by Δ_B. $C_{(s)}^{l\beta}\omega_{(s)}^{l} = \left(A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta}\right)\hat{A}_{(s)}^{l} = A_{(s)}^{l\beta}\hat{A}_{(s)}^{l} + \tilde{C}_{(s)}^{l\beta}\hat{A}_{(s)}^{l}$. Evaluate the summands: $\hat{A}_{(s)}^{l} = \sum_\beta A_{(s)}^{l\beta}$ implies (for 0/1 variables) that $A_{(s)}^{l\beta}\hat{A}_{(s)}^{l} = A_{(s)}^{l\beta}$, and $\tilde{C}_{(s)}^{l\beta}\hat{A}_{(s)}^{l} \le \hat{A}_{(s)}^{l}\sum_\beta \tilde{C}_{(s)}^{l\beta} \le \hat{A}_s^{l}\left(\hat{A}_{(s)}^{l-1} - \hat{A}_{(s)}^{l}\right) = \hat{A}_{(s)}^{l} - \hat{A}_{(s)}^{l} = 0$, since $\hat{A}_{(s)}^{l} \le \hat{A}_{(s)}^{l} + \sum_\beta \tilde{C}_{(s)}^{l\beta} \le \hat{A}_{(s)}^{l-1}$. Substituting, we find $A_{(s)}^{l\beta} = C_{(s)}^{l\beta}\omega_{(s)}^{l}$ as desired. Likewise $C_{(s)}^{l\beta}\left(1-\omega_{(s)}^{l}\right) = \left(A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta}\right)\left(1-\hat{A}_{(s)}^{l}\right) = \tilde{C}_{(s)}^{l\beta} - \tilde{C}_{(s)}^{l\beta}\hat{A}_{(s)}^{l} + A_{(s)}^{l\beta}\left(1-\hat{A}_{(s)}^{l}\right) = \tilde{C}_{(s)}^{l\beta} - 0 + 0$, as desired.

Finally we must translate the objective, assuming $C_C \wedge \Delta_F \wedge C_A \wedge \Delta_B$. Using the calculations in the proof above, we can work out

$$
\begin{aligned}
C_{s_1\ldots s_l}^{l\beta} C_{s_1\ldots s_l}^{l-1\alpha}\left(\prod_{k=1}^{l}\omega^{k}{}_{s_1\ldots s_l}\right) &= C_{s_1\ldots s_l}^{l\beta} C_{s_1\ldots s_l}^{l-1\alpha}\omega_{(s)}^{l} \\
&= \left(A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta}\right)\left(A_{(s)}^{l-1\alpha} + \tilde{C}_{(s)}^{l-1\alpha}\right)\hat{A}_{(s)}^{l} \\
&= A_s^{l\beta}A_s^{l-1\alpha}\hat{A}_s^{l} + \hat{A}_s^{l}\tilde{C}_s^{l\beta}A_s^{l-1\alpha} + \hat{A}_{(s)}^{l}\hat{A}_{(s)}^{l-1}\tilde{C}_{(s)}^{l-1\alpha}\left(A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta}\right) \\
&= A_s^{l\beta}A_s^{l-1\alpha}\hat{A}_s^{l} + 0A_s^{l-1\alpha} + \hat{A}_{(s)}^{l}0\left(A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta}\right) \\
&= A_s^{l\beta}A_s^{l-1\alpha}
\end{aligned}
$$

Substituting this result into equation (1), after using $\left(\prod_{k=1}^{l}\omega^{k}{}_{s_1\ldots s_l}\right)\omega_{(s)}^{l} = \prod_{k=1}^{l}\omega^{k}{}_{s_1\ldots s_l}$ to simplify the μ term, yields equation (5) as desired.

Appendix II

We now prove the validity, according to the criterion of equation (3) with temperature *T*=0, of the change of variable (9),(10) which adds dependent permutation variables *P* at each level in the hierarchy, and also the change of variable (13) which completes the derivation of Frameville.

First we show the validity of the change of variables (9), (10), from C_A (equation (6)) to $C_{A,P}$ (equation (12)). For the most part, this change of variables just consists of adding and removing auxiliary variables, i.e. variables in *X\x* or*Y\y*, in the notation of section 3.0. These changes would affect the entropy terms (omitted in this paper) in the free energy expressions at nonzero temperature. In C_A, which is mapped to $C_{A,P}$ via Δ_F, the auxiliary variables are the $\tilde{C}$'s. In $C_{A,P}$, which is mapped back to C_A via Δ_B, the auxiliary variables are those x_j^l and P_j^l for which $l < L$, together with alloc_j^l. Most of the constraints on nonauxiliary variables are explicitly preserved between C_A and $C_{A,P}$. There are only a few details to check.

In the forward direction, it is not explicit in C_A that $\hat{A}_{(s)}^{l} \le \hat{A}_{(s)}^{l-1}$. It is, however, implicit: $A_{(s)}^{l\beta} \le A_{(s)}^{l\beta} + \tilde{C}_{(s)}^{l\beta} \le \sum_\alpha \mathrm{INA}_{\alpha s_l \beta} A_{(s)}^{l-1\alpha} \le \sum_\alpha\left(A_s^{l-1\alpha}\right) = \hat{A}_{(s)}^{l-1}$; but since$A_s^{l\beta} \in \{0,1\}$ and $\hat{A}_{(s)}^{l} = \sum_\beta A_{(s)}^{l\beta} \in \{0,1\}$, we have $\hat{A}_{(s)}^{l} = \max_\beta A_s^{l\beta} \le \max_\beta \hat{A}_{(s)}^{l-1} = \hat{A}_{(s)}^{l-1}$. The expansion for alloc_k^L in $C_{A,P}$ in terms of Heaviside Θ functions just says alloc=1 if *k* is in {1, 2, ... $k_{\max}$}, otherwise alloc=0. This just names the domain of allowed *k*'s in C_A's constraints on *x* and *P*.

We must also check that the required auxiliary variables exist, i.e. that their constraints are consistent. The auxiliary *x*'s are uniquely determined by the regular *x*'s and the auxiliary *P*'s, which are in turn constrained by permutation-like constraints which can be satisfied in many ways. The $\tilde{C}$ variables are constrained by Δ_B to be less than or equal to quantities which are guaranteed by $C_{A,P}$ to be nonnegative, so that at least the configuration $\tilde{C} = 0$ is consistent.

17. E. Mjolsness. Connectionist grammars for high-level vision, in *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, eds. Vasant Honavar and Leonard Uhr, Academic Press, 1994.

18. E. Mjolsness, G. Gindi, and P. Anandan. Optimization in model matching and perceptual organization. Neural Computation 1(2), 1989.

19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, 1988.

20. C. Peterson and B. Soderberg. A new method for mapping optimization problems onto neural networks. International Journal of Neural Systems, 1(3), 1989.

21. G. Pinkas. Energy minimization and the satisfiability of propositional calculus. Neural Computation 3(2), 1991.

22. T. Plate. Distributed representations and nested compositional structures. PhD thesis, Department of Computer Science, University of Toronto, 1994.

23. P. Prusinkiewicz. *The Algorithmic Beauty of Plants*. Springer-Verlag New york, 1990.

24. E. Saund. A multiple cause mixture model for unsupervised learning. Neural Computation, v. 7, n. 1, 1995.

25. A. Stolcke. Unification as Constraint Satisfaction in Structured Connectionist Networks.

26. D. E. Van den Bout and T. K. Miller, III. Graph partitioning using annealed networks. IEEE Transactions on Neural Networks, 1(2):192-203, 1990.

Appendix I

We prove the validity, according to the criterion of equation (3) with temperature T=0, of the change of variable (4) from rule choice variables C to term aliveness variables A..

First we show that the constraints on C (equation (2)), and the forward change of variables Δ_F (equation (4)), imply the constraints on A (equation (6)), i.e. $C_C \wedge \Delta_F \Rightarrow C_A$.

Clearly Δ_F preserves the 0/1-valuedness of all the variables. The condition on $A^{0,\alpha}$ is unchanged in C_A from C_C. From $C_C \wedge \Delta_F$ we can calculate $\sum_\beta A^{l\beta}_{(s)} = \omega^l_{(s)} \sum_\beta C^{l\beta}_{(s)} \geq \omega^l_{(s)} \omega^l_{(s)} = \omega^l_{(s)} = \hat{A}^l_{(s)}$; but $\omega^l_{(s)} \sum_\beta C^{l\beta}_{(s)} \leq \omega^l_{(s)} \prod_{k=1}^{l-1} \omega^k_{(s)}$ so $\sum_\beta A^{l\beta}_{(s)} \leq \omega^l_{(s)} = \hat{A}^l_{(s)}$, and we have proven two inequalities implying $\sum_\beta A^{l\beta}_{(s)} = \hat{A}^l_{(s)}$. Note also $\prod_{k=1}^{l} \omega^k_{(s)} \leq \omega^l_{(s)} = \omega^l_{(s)} \omega^l_{(s)} \leq \omega^l_{(s)} \prod_{k=1}^{l-1} \omega^k_{(s)}$, so $\omega^l_{(s)} = \prod_{k=1}^{l} \omega^k_{(s)}$.

The constraints on x and P are direct translations under Δ_F, from C_C to C_A. So we only have to derive the two inequalities on $\tilde{C}$ and A, in C_A. We can calculate $\hat{A}^l_{(s)} + \sum_\beta \tilde{C}^{l\beta}_{(s)} = \sum_\beta \left(A^{l\beta}_{(s)} + \tilde{C}^{l\beta}_{(s)} \right) = \sum_\beta C^{l\beta}_{(s)} \left[\omega^l_{(s)} + 1 - \omega^l_{(s)} \right] = \sum_\beta C^{l\beta}_{(s)}$. But $\sum_\beta C^{l\beta}_{(s)} \leq \prod_{k=1}^{l-1} \omega^k_{(s)}$, which equals $\omega^{l-1}_{(s)}$ for $l \geq 2$ (otherwise l=1 and $\prod = 1 = \hat{A}^0$), which is $\hat{A}^{l-1}_{(s)}$ by Δ_F. This establishes the first inequality. For the second inequality, calculate $A^{l\beta}_{(s)} + \tilde{C}^{l\beta}_{(s)} = C^{l\beta}_{(s)} \left[\omega^l_{(s)} + 1 - \omega^l_{(s)} \right] = C^{l\beta}_{(s)}$, which is $\leq \sum_\alpha \mathrm{INA}_{\alpha s_l \beta} C^{l-1\alpha}_{(s)}$, as required.

So $C_C \wedge \Delta_F \Rightarrow C_A$. Next we must show $C_C \wedge \Delta_F \Rightarrow \Delta_B$. But Δ_B follows trivially from Δ_F alone.

Now we must show that the constraints on A (equation (6)), and the backward change of variables Δ_B (equation (4)), imply the constraints on C (equation (2)), i.e. $C_A \wedge \Delta_B \Rightarrow C_C$. Clearly Δ_B preserves the 0/1-valuedness of the ω and P variables. However, the C variables could also take on the value 1+1=2, except that $C^{l\beta}_{(s)} \leq \sum_\beta C^{l\beta}_{(s)} = \hat{A}^l_{(s)} + \sum_\beta \tilde{C}^{l\beta}_{(s)} \leq \hat{A}^{l-1}_{(s)} \leq 1$.

The constraints on $\sum_\beta C^{l\beta}_{(s)}$ in C_C may be verified as follows. From Δ_B, $\sum_\beta C^{l\beta}_{(s)} = \hat{A}^l_{(s)} + \sum_\beta \tilde{C}^{l\beta}_{(s)} \geq \hat{A}^l_{(s)} = \omega^l_{(s)}$. On the other hand we have already calculated that $\sum_\beta C^{l\beta}_{(s)} = \hat{A}^l_{(s)} + \sum_\beta \tilde{C}^{l\beta}_{(s)} \leq \hat{A}^{l-1}_{(s)}$, which is ≤ 1 for l=1, and $= \omega^{l-1}_{(s)}$ for l=2 (as required for $\sum_\beta C^{l\beta}_{(s)} \leq \prod_{k=1}^{l-1} \omega^k_{(s)}$) and also for all larger values of l. But by induction on l, for $l \geq 2$ we have

simple enough one may derive algorithms close to ordinary neural networks, including learning [3]. Otherwise, we find mixed logical and numerical processing in a more elaborate architecture. The grammar and therefore the network architecture can be tuned to specific domains (such as high-level vision) by including domain-specific grammar rules (such as the geometric invariances of objects [16]) which get hard-wired into the resulting neural networks. For highly expressive architectures one generally cannot claim more or less complete expressiveness, only more or less appropriate expressiveness. Further work on the stochastic grammar approach will determine whether such claims should be made about it.

Acknowledgment. Gary Cottrell provided many useful comments in the preparation of this paper.

References

1. P. Anandan, S. Letovsky, and E. Mjolsness. Connectionist variable-binding by optimization. Proceedings of the Cognitive Science conference, August 1989.

2. P. Baldi and Y. Chauvin. Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation*, in press.

3. B. Cheng and D. M. Titterington. Neural networks: A review from a statistical perspective. Statistical Sciences, v. 9 n. 1, pp. 2-54, 1994.

4. M. Derthick. Mundane Reasoning by Parallel Constraint Satisfaction. Morgan-Kaufmann 1990.

5. S. Gold, A. Rangarajan, and E. Mjolsness. Learning with preknowledge: Clustering with point and graph matching distance measures. *Neural Computation*, in press. Preliminary version in: *Neural Information Processing Systems 7*, MIT Press 1995.

6. S. Gold, C. P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. In: *Neural Information Processing Systems 7*, MIT Press 1995.

7. G. Hinton and R. Zemel. Autoencoders, minimum description length and Helmholtz free energy. *Advances in Neural Information Processing Systems 6*, eds. J. Cowan, G. Tesauro, and J. Alspector, Morgan Kauffman 1994.

8. J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences USA, v. 81, p. 3088-3092, 1984.

9. J. J. Kosowsky and A. L. Yuille, The invisible hand algorithm: Solving the assignment problem with statistical physics. Neural Networks 7(3):477-490, 1994.

10. T. Lange,. and M. G. Dyer. Dynamic, non-local role-bindings and inferencing in a localist network for natural language understanding. In David S. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, pp. 545-552. Morgan-Kaufmann, 1988.

11. C. von der Malsburg and E. Bienenstock. Statistical coding and short-term synaptic plasticity: A scheme for knowledge representation in the brain. In *Disordered Systems and Biological Organization*, 247-252, Springer-Verlag 1986.

12. E. Mjolsness and C. Garrett. Algebraic transformations of objective functions. Neural Networks, 3:651-669, 1990.

13. E. Mjolsness, C. Garrett, and W. Miranker. Multiscale optimization in neural networks, IEEE Transactions on Neural Networks, vol 2 no 2, March 1991.

14. E. Mjolsness and W. Miranker. Greedy Lagrangians for Neural Networks: Three Levels of Optimization in Relaxation Dynamics. Technical Report YALEU/DCS/TR-945, Yale Computer Science Department, January 1993.

15. E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. Journal of Theoretical Biology, 152(4):429-454, 1991.

16. E. Mjolsness. Bayesian inference on visual grammars by neural nets that optimize. Technical Report YALEU/DCS/TR-854, Yale Computer Science Department, May 1991. Available on neuroprose and at URL http://www-cse.ucsd.edu/users/emj .

distribution, and then generate a number of scenes (rather than just one as above) which can be used as input to a learning algorithm. The resulting constrained optimization problem for learning the model base parameters *u* could for example be:

$$E_{\text{learn}}(u, \mu, \{M, \text{ina}, x, \text{alloc}\}; \text{INA}) = \sum_p E_{\text{recog}}\left(M^p, \text{ina}^p, x^p, \text{alloc}^p; u, \mu, \text{INA}\right) + \frac{1}{2\sigma_u^2}\sum_{\alpha\sigma\beta}\left[u^{\alpha\sigma\beta}\right]^2 + \frac{1}{2\sigma_u^2}\sum_{\sigma\beta}\left[\mu^{\sigma\beta}\right]^2 + \tilde{\mu}\sum_{\alpha\sigma\beta}\text{INA}_{\alpha\sigma\beta} \tag{16}$$

if the prior on *u* and μ is taken to be Gaussian and that on INA is taken to be independent on each two-valued matrix entry, subject to the INA constraint. This type of learning problem has been solved efficiently with neural networks for the simpler point-matching problems that arise in nonhierarchical generative grammar models [5].

4.0 Discussion

In this version of the Frameville architecture, by comparison with earlier versions [18][17], we can see several advantages and limitations. The depth of a derivation is bounded by the arbitrary integer *L*. Any number of "distractor" or noise objects can be modelled as parts of a level-one "background" object. The model-side compositional hierarchy $\text{INA}_{\alpha\beta} = \sum_\sigma \text{INA}_{\alpha\sigma\beta}$ is a general graph, rather than being limited to a tree or a DAG. The stochastic grammar can be extended to define model-learning as another constrained optimization problem, in which the bias of the learning algorithm is explicitly given by a prior probability distribution on model bases in the compound grammar rather than implicitly given by a neural net wiring diagram. And as before, the number of required variables grows as a low power of the size of the data, and the architecture can be diagrammed with a 3D embedding that suggests control flow strategies.

On the other hand, the Frameville architecture presented here is missing several crucial link types (such as explicit ISA, EQU for equality of frame instances, SIB for siblings in the composition hierarchy, and so on); the data-side dynamic *ina* links are restricted to form a tree (a restriction removed by EQU for translating predicates in [1]; see also [15]); the remaining slot indices σ are inconvenient for large collection objects; all variables remain indexed by their level numbers in the grammar derivation; and we have not exhibited a neural dynamics, or other global constrained optimization algorithm, which can solve the Frameville optimization problems in a scalable way as has been done for less difficult optimization problems.

4.1 Comparison with Other Integration Schemes

Bayesian networks [19] are analogous to derivations of our grammar, in that they have conditionally independent probability distributions forming an underlying stochastic domain model, they can be learned, and they provide a high-level language for integrating multiple process models. They have the important expressive advantage that they are DAGs, rather than trees as our derivations are; so they are "multiple cause models" [24]. We have formulated context-sensitive deterministic L-system grammars whose derivations are DAGs, for use in biology [15], and a similar generalization may be possible here. But in a Bayesian net each node's probability distribution tends to be much less powerful than our Boltzmann distributions, and there is no grammar inducing structure or regularity on the derivation tree. Also there is no analog of the "correspondence" or "variable-binding" problem introduced by relabelling the data according to the random permutation matrix *P*, which speeds up the computations but limits the expressive power of the Bayesian networks.A related trend in machine learning is the introduction

A related trend in machine learning is the introduction and formalization of appropriate "structure" in a model class by model-generation procedures. A Minimum Description Length approach to neural net learning [7] and Hidden Markov Model-generating neural networks [2] are recent examples. Our grammar schemata may be viewed as procedures for generating models, with appropriate structure, in the following model classes: (1) larger parameterized L-system grammars; (2) simplified semantic networks; (3) Bayesian net-like grammar derivations; (4) constrained optimization problem formulations; and (5) relaxation-based neural networks. A stochastic grammar formalism may be especially expressive in this role.

There are a number of neural network architectures which do variable-binding [4][10][21][25], some of which are relaxation-based networks which pose constrained optimization problems. Some of our work on transformations from optimization problems to neural networks could be applied to those systems as well. The stochastic grammar approach avoids the common problem of creating slightly extended implementations of symbolic processing in neural networks, rather than true integration of the advantages of each, by starting with an underlying statistical domain model. From this model emerge domain-specific "distance" or "similarity" measures for pattern recognition and learning on structured objects. If a statistical domain model is

$$E(A,x) = H_0\left(x^0\right) + \sum_{l=1}^{L} \sum_{\alpha\sigma\beta} \sum_{ij} \mathrm{INA}_{\alpha\sigma\beta} \mathrm{ina}_{i\sigma j}^{l} M_{\alpha i}^{l-1} M_{\beta j}^{l} \left[H^{(\alpha\sigma\beta)}\left(x_j^{l}, x_i^{l-1}, u^{(\alpha\sigma\beta)}\right) - \mu^{(\sigma\beta)} \right] \tag{14}$$

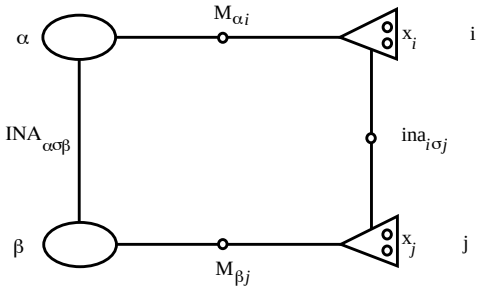

Figure 4: Frameville objective.

This objective is of polynomial degree five or a little more depending on H. Further reduction down to degree three is possible as in [12]. The constraints for the (*M*, ina, alloc, *x*) variables are:

$$\begin{aligned} & M, \mathrm{ina}, \mathrm{alloc}, \mathrm{INA} \in \{0,1\} \\ & \sum_i M_{\alpha i}^{0} = \delta_{\alpha,\,\mathrm{root}} \quad \wedge \quad x_k^{L} = x_k \quad \wedge \quad \mathrm{alloc}_k^{L} = \Theta\left(k+\tfrac{1}{2}\right)\Theta\left(k_{\max}-k+\tfrac{1}{2}\right) \\ & \sum_\beta M_{\beta j}^{l} = \mathrm{alloc}_j^{l} \\ & \sum_{i\sigma} \mathrm{ina}_{i\sigma j}^{l} = \mathrm{alloc}_j^{l} \\ & \sum_j \mathrm{ina}_{i\sigma j}^{l} \le \mathrm{alloc}_i^{l-1} \\ & \sum_j \mathrm{ina}_{i\sigma j}^{l} M_{\beta j}^{l} \le \sum_\alpha \mathrm{INA}_{\alpha\sigma\beta} M_{\alpha i}^{l-1} \end{aligned} \tag{15}$$

The four constraint cliques relating different kinds of variables are diagrammed as before, in Figure 5. Again, the validity of

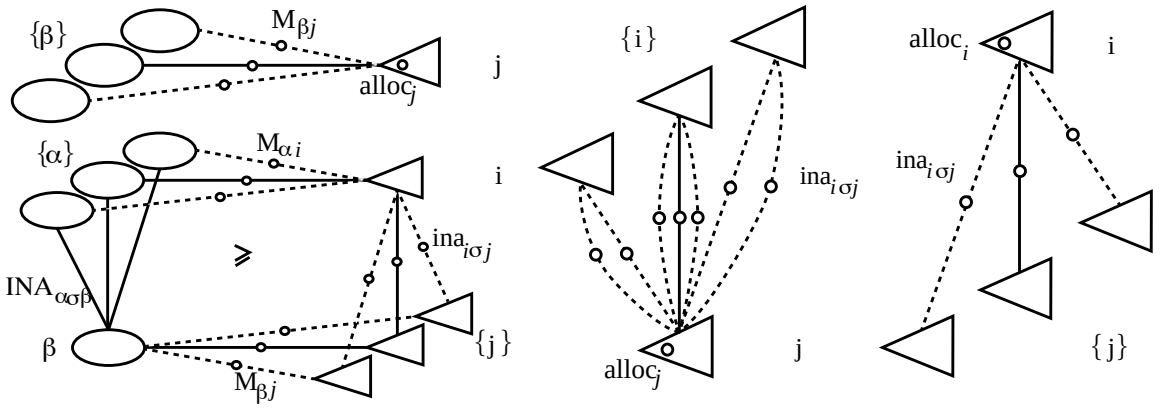

Figure 5: Frameville constraints

this change of variables is proven in Appendix II. Note that the $\sum_j \mathrm{ina}_{i\sigma j}^{l}$ inequalities reflect the possibility of object deletions in the original grammar, which has not been hidden in this particular problem formulation.

An earlier version of Frameville [18], [16], which did not have the slot index σ on INA or *ina* and which specialized INA by restricting it to be a tree, can be derived (for tree INA's) from this version by a further change of variables which we omit. A version including slot indices was related to predicate calculus in [1]. Related symbolic optimization architectures include [11], [21],[25].

3.4.1 Learning in Frameville

Important aspects of the model base such as the constant parameters *u* and μ which tune the consistency function *H*, or the INA matrix which specifies the equivalent detailed grammar, may be learned by starting off the recursive object-generation grammar with extra "metagrammar" or "compound grammar" rules which first choose *u* and/or INA from a prior probability

The new objective function is:

$$E(A, x) = H_0\left(x^0\right) + \sum_{l=1}^{L} \sum_{s} \sum_{\alpha\beta} \sum_{ij} \mathrm{INA}_{\alpha s_l \beta} A^{\,l\ \beta}_{s_1 \ldots s_l} A^{\,l-1\alpha}_{s_1 \ldots s_l} P^{\,l}_{s_1 \ldots s_l, j} P^{\,l-1}_{s_1 \ldots s_{l-1}, i} \left[H^{(\alpha s_l \beta)}\left(x^{\,l}_j, x^{\,l-1}_i, u^{(\alpha s_l \beta)}\right) - \mu^{(s_l \beta)} \right] \tag{11}$$

The constraints for the (*A*, *P*, alloc, *x*) variables are:

$$\begin{aligned} & A, \hat{A}, P, \mathrm{alloc}, \mathrm{INA} \in \{0, 1\} \\ & A^{0,\alpha} = \delta_{\alpha,\,\mathrm{root}} \quad \wedge \quad x^{\,L}_k = x_k \quad \wedge \quad \mathrm{alloc}^{\,L}_k = \Theta\left(k + \frac{1}{2}\right)\Theta\left(k_{\max} - k + \frac{1}{2}\right) \\ & \hat{A}^{\,l}_{(s)} = \sum_{\beta} A^{\,l\ \beta}_{(s)} \quad \wedge \quad \hat{A}^{\,l}_{(s)} \le \hat{A}^{\,l-1}_{(s)} \\ & A^{\,l\ \beta}_{s_1 \ldots s_l} \le \sum_{\alpha} \mathrm{INA}_{\alpha s_l \beta} A^{\,l-1\alpha}_{s_1 \ldots s_{l-1}} \\ & \sum_{(s)} P^{\,l}_{(s)\,j} = \mathrm{alloc}^{\,l}_j \quad \wedge \quad \sum_j P^{\,l}_{(s)\,j} = \hat{A}^{\,l}_s \end{aligned} \tag{12}$$

The validity of this change of variables is proven in Appendix II.

3.3.2 2nd Change: Eliminate Path Indices

Now we eliminate all the exponentially growing derivation path indices *(s)* by eliminating *A* and *P*. The change of variables is:

(13)

Δ_F:

$$\begin{aligned} & x^{\,l} = x^{\,l} \quad \wedge \quad \mathrm{alloc}^{\,l}_j = \mathrm{alloc}^{\,l}_j \\ & M^{\,l}_{\beta j} = \sum_{(s)} A^{\,l\ \beta}_{(s)} P^{\,l}_{(s)\,j} \\ & \mathrm{ina}^{\,l}_{i s_l j} = \sum_{s_1 \ldots s_{l-1}} P^{\,l-1}_{s_1 \ldots s_{l-1}, i} \quad P^{\,l}_{s_1 \ldots s_l, j} \end{aligned}$$

Δ_B:

$$\begin{aligned} & x^{\,l} = x^{\,l} \quad \wedge \quad \mathrm{alloc}^{\,l}_j = \mathrm{alloc}^{\,l}_j \\ & A^{\,l\ \beta}_{(s)} = \sum_j M^{\,l}_{\beta j} P^{\,l}_{(s)\,j} \quad \wedge \quad \hat{A}^{\,l}_{(s)} = \sum_{\beta} A^{\,l\ \beta}_{(s)} \\ & P^{\,0}_{(\ )\,i} = M^{0}_{\mathrm{root}\,i} \\ & P^{\,l}_{s_1 \ldots s_l, j} = \sum_i \mathrm{ina}^{\,l}_{i s_l j} P^{\,l-1}_{s_1 \ldots s_{l-1}, i} \end{aligned}$$

This is quadratic in the link variables A, P, M, and ina.

3.4 Frameville with Slots

The resulting objective function and clique diagrams are shown below:

3.3 Eliminate Derivation Indices

The *(s)* indices still introduce a rapidly, perhaps exponentially growing number of variables, corresponding to all the possible derivations in the grammar. Two further changes of variable can eliminate the excess variables. First, we introduce auxiliary variables *P* and *x* at all levels, not just *l*=*L*, and we eliminate the remaining choice variables $\tilde{C}$. This step introduces trivial dependent "frame instances" in correspondence with the derivation objects. Second, we eliminate the derivation objects and their aliveness variables *A* in favor of the frame instances and relationships between them and between frames (models) and their instances. These changes are diagrammed in figure 3, using triangles to represent frame instances and showing all relevant variables in two neighboring cliques at levels (*L*-2, *L*-1) and (*L*-1, *L*). The dotted lines show the introduction of the new

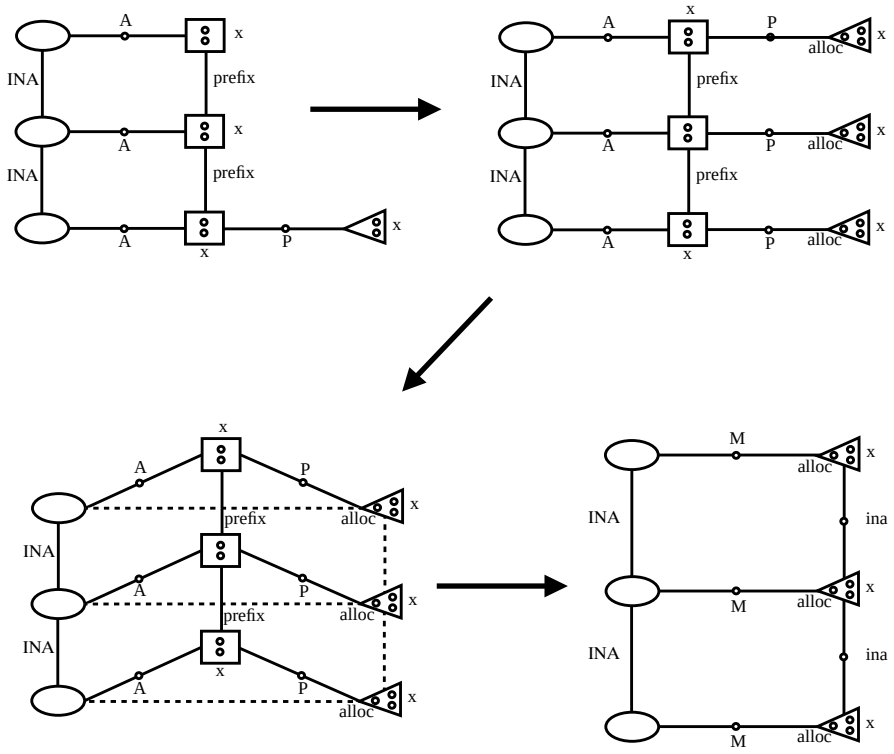

Figure 3: Change of variables to Frameville architecture.

link types: the match or instantiation variables "*M*" between frame models and instances, and the Part-of variables "ina" between instances.

3.3.1 1st Change: Add Instances

The change of variables is:

$$
\begin{aligned}
& A = A \quad \wedge \quad \hat{A} = \hat{A} \quad \wedge \quad P^L = P \quad \wedge \quad x^L = x^L \\
\Delta_F: \quad & \text{auxiliary variables: } x^l_j \in \Re \quad \wedge P^l_{(s)j} \in \{0,1\} \quad (l < L),\ \text{alloc}^l_j \in \{0,1\} \\
& x^l_j = \textstyle\sum_{(s)} P^l_{(s)j}\ x^l_{(s)} \quad \wedge \quad \sum_{(s)} P^l_{(s)j} = \text{alloc}^l_j \quad \wedge \quad \sum_j P^l_{(s)j} = \hat{A}^l_{(s)}
\end{aligned}
\tag{9}
$$

and inversely

$$
\begin{aligned}
& A = A \quad \wedge \quad \hat{A} = \hat{A} \quad \wedge \quad P = P^L \quad \wedge \quad x^L = x^L \\
& x^l_{(s)} = \textstyle\sum_j P^l_{(s)j}\ x^l_j \\
{}_B: \quad & \tilde{C}^{\,l\,\beta}_{(s)} \in \{0,1\} \ \wedge \quad \textstyle\sum_\beta \tilde{C}^{\,l\,\beta}_{(s)} \le \hat{A}^{l-1}_{(s)} - \hat{A}^l_{(s)} \\
& \text{auxiliary variables:} \\
& \tilde{C}^{\,l\,\beta}_{(s)} \le \textstyle\sum_\alpha \text{INA}_{\alpha s_l \beta} A^{l-1\alpha}_{(s)} - A^{l\beta}_{(s)}
\end{aligned}
\tag{10}
$$

3.1.1 Specialization to Point-Matching

We can make contact with previous experiments on nonhierarchical point-matching by specializing the aliveness variable formulation of the constrained optimization problem. If there is only one model accessible after L levels of derivation, say the "point" model, then $A^{L\beta}_{(s)} = \delta_{\beta,\,\text{point}}\hat{A}^{L}_{(s)}$ and we can substitute $x_{(s)} = \sum_{(s)} P_{(s)k} x^{L}_{k}$ into the l=L summand of the objective and simplify. By specializing H to include translational an rotational degrees of freedom at every node in the hierarchy, we arrive at the objective

$$\begin{aligned} E(A,x) \;=\; H_0\left(x^0\right) + \sum_{l=1}^{L-1}\sum_{(s)}\sum_{\alpha\beta} A^{\,l\,\beta}_{s_1\ldots s_l} A^{\,l-1\alpha}_{s_1\ldots s_{l-1}} \left[\frac{1}{2\sigma_l^2}\left\| x^{\,l}_{s_1\ldots s_l} - x^{\,l-1}_{s_1\ldots s_{l-1}} - \Omega^{\,l-1}_{s_1\ldots s_{l-1}}\cdot y^{\,\alpha s_l\beta}\right\|^2 + G\left(\Omega^{\,l-1}_{s_1\ldots s_{l-1}},\,\sigma_l,\,\sigma_{l-1}\right) - \mu^{(s_l\,\beta)}\right] \\ + \sum_{k}\sum_{(s)}\sum_{\alpha} A^{\,L-1\alpha}_{s_1\ldots s_{l-1}}\; P_{(s)k}\left[\frac{1}{2\sigma_L^2}\left\| x^{\,L}_{k} - x^{\,L-1}_{s_1\ldots s_{l-1}} - \Omega^{\,L-1}_{s_1\ldots s_{l-1}}\cdot y^{\,\alpha s_l\,\text{point}}\right\|^2 + G\left(\Omega^{\,L-1}_{s_1\ldots s_{l-1}},\,\sigma_L,\,\sigma_{L-1}\right) - \mu^{(s_l\,\text{point})}\right] \end{aligned} \tag{7}$$

and the constraints

$$\begin{aligned} &A, \hat{A}, P, \text{INA} \in \{0,1\} \\ &A^{0,\alpha} = \delta_{\alpha,\,\text{root}} \\ &\hat{A}^{\,l}_{(s)} = \sum_{\beta} A^{\,l\,\beta}_{(s)} \\ &A^{\,l\,\beta}_{s_1\ldots s_l} \le \sum_{\alpha} \text{INA}_{\alpha s_l \beta} A^{\,l-1\alpha}_{s_1\ldots s_{l-1}} \\ &\sum_{\beta} \text{INA}_{\alpha\sigma\beta} \le 1 \\ &\sum_{(s)} P_{(s)k} = \Theta\left(k+\frac{1}{2}\right)\Theta\left(k_{\max}-k+\frac{1}{2}\right) \quad\wedge\quad \sum_{k} P_{(s)k} = \hat{A}^{\,L}_{(s)} \end{aligned} \tag{8}$$

(Here Θ is the 0/1-valued Heaviside function, used to constrain k to be between 1 and its maximum.) This constrained optimization problem is a hierarchical generalization of a previous nonhierarchical (L=1 or 2) point-matching architecture implemented in [6]:

$$E(m,A,t,\mu,\nu) \;=\; \frac{1}{2\sigma^2}\sum_{ij} m_{ij}\left\|x_i - t - Ay_j\right\|^2 - \alpha\sum_{ij} m_{ij} - g(A)$$

with constraints $m \in \{0,1\} \quad\wedge\quad \sum_i m_{ij} \le 1 \quad\wedge\quad \sum_j m_{ij} \le 1$. Here g is a regularizer on the affine transformation matrix A, and G is a similar regularizer for the affine transformation Ω between levels in the hierarchical object grammar, parameterized by the noise parameters σ. In particular if Ω is regularized to favor rotations, then we obtain a mobile-like hierarchical object model.

3.2 Mapping to Neural Networks

An objective function such as equation (5), and its associated constraints (e.g. that each A is zero or one), can be mapped into an analog neural network by a variety of techniques [8][9] which require that we expand H into a polynomial, and allocate multi-way synapses for each monomial in the resultant polynomial expression for E. The constraints can be treated similarly. Monomials of degree two such as $3xy$ are mapped to a pair of conventional synapses from neuron x to neuron y and back. If the polynomials have a regular structure, the synapse count may be considerably reduced by introducing intermediate variables as in [12]. Convergence can often be greatly accelerated by introducing clocked dynamics as in [14]. The resulting networks will have a localist encoding because of the 0/1-valued A and P variables, which is an inefficient use of space since at least the permutation matrix P is sparse. It is possible that new mappings to localist [12] or distributed [22] representations will eventually allow the same model to run with much less circuitry. Effective use of deterministic annealing methods can also improve the quality of the local minima to which such algorithms converge, for both recognition and learning [6][5].

3.1 Degree Reduction

If we change to the object "aliveness" variables $A^{l\,\beta}_{s_1\ldots s_l}$ [15] which are 1 or 0 depending on whether an object with path index *(s)* and type α is in the derivation or not, rather than the "rule choice" variables $C^{l\,\beta}_{s_1\ldots s_l}$ and $\omega^{l}_{(s)}$, we can reduce the objective and constraints to low polynomial degree. To this end, we use change of variable equations like $A^{l\,\beta}_{(s)} = C^{l\,\beta}_{(s)}\ \omega^{l}_{(s)}$, which says that an object is live with type β if and only if it was a preobject with type β and it survived to become an object. For deleted objects, however, we retain the choice variables $\tilde{C}$. In detail, the change of variables is:

$$
\begin{aligned}
\Delta_F\text{:}\quad & A^{0,\alpha} = \delta_{\alpha,\,\text{scene}} \quad\wedge\quad \hat{A}^{0} = 1 \quad\quad // \ (\delta \text{ is the Kronecker delta function}) \\
& A^{l\,\beta}_{(s)} = C^{l\,\beta}_{(s)}\ \omega^{l}_{(s)} \quad\wedge\quad \hat{A}^{l}_{(s)} = \omega^{l}_{(s)} \quad\wedge\quad \tilde{C}^{l\,\beta}_{(s)} = C^{l\,\beta}_{(s)}\left(1-\omega^{l}_{(s)}\right) \quad (l \geq 1) \\
\Delta_B\text{:}\quad & C^{l\,\beta}_{(s)} = A^{l\,\beta}_{(s)} + \tilde{C}^{l\,\beta}_{(s)} \quad\wedge\quad \omega^{l}_{(s)} = \hat{A}^{l}_{(s)} \quad (l \geq 1)
\end{aligned}
\tag{4}
$$

The new objective function is:

$$
E(A,x) = H_0\left(x^0\right) + \sum_{l=1}^{L}\sum_{(s)}\sum_{\alpha\beta} A^{\,l\,\beta}_{s_1\ldots s_l} A^{\,l-1\,\alpha}_{s_1\ldots s_{l-1}} \left[H^{(\alpha s_l\ \beta)}\left(x^{\,l}_{s_1\ldots s_l}, x^{\,l-1}_{s_1\ldots s_{l-1}}, u^{(\alpha s_l \beta)}\right) - \mu^{(s_l\,\beta)}\right] \tag{5}
$$

which is a sum of low-degree monomial interaction terms. (In a Hopfield neural network, a quadratic monomial becomes two regular synapses and a cubic monomial becomes three multiplicative synapses.) The *H* functions may be interpreted as context-dependent "distance" or "similarity" measures between a model β and the set of instantiation parameters $x^{\,l}_{s_1\ldots s_l}$ ultimately constrained by data x_k. The associated constraints for the (*A*, *P*, *x*) variables are:

$$
\begin{aligned}
A, \hat{A}, \tilde{C}, P, \text{INA} &\in \{0,1\} \\
A^{0,\alpha} &= \delta_{\alpha,\,\text{root}} \\
\hat{A}^{l}_{(s)} &= \sum\nolimits_{\beta} A^{\,l\,\beta}_{(s)} \\
\hat{A}^{\,l}_{s_1\ldots s_l} + \sum\nolimits_{\beta} \tilde{C}^{\,l\,\beta}_{s_1\ldots s_l} &\leq \hat{A}^{\,l-1}_{s_1\ldots s_{l-1}} \\
A^{\,l\,\beta}_{s_1\ldots s_l} + \tilde{C}^{\,l\,\beta}_{s_1\ldots s_l} &\leq \sum\nolimits_{\alpha} \text{INA}_{\alpha s_l \beta} A^{\,l-1\,\alpha}_{s_1\ldots s_{l-1}} \\
x_k = \sum\nolimits_{(s)} P_{(s)\,k} x_{(s)} \quad\wedge\quad \sum\nolimits_{(s)} P_{(s)\,k} = 1 \quad\wedge\quad \sum\nolimits_{k} P_{(s)\,k} &= \hat{A}^{\,L}_{(s)}
\end{aligned}
\tag{6}
$$

For these objectives, constraints, and forward and backward changes of variable, the change of variable conditions () are proven to hold in Appendix I. The advantage of the new system is that all polynomial degrees are low (3 or 4 rather than *L*). However, the overly numerous *(s)*-indexed variables remain.

The objective and constraints of (5) and (6) may be diagrammed as shown below, in which the small open circles are variables, the ovals represent the models (which store constant database infor mation like *u*, μ, *H*, and INA), and the squares are possible terms in a derivation. Each diagram below shows only a representative clique of related variables which interact in one constraint in a conjunction thereof or one summand of the objective function. Links connect variables which share an index. The objective function and the inequality constraint are shown in these two diagrams, with object types on the left and instances (grammatical "object" terms and their associated parameters, *x*) on the right of each diagram:

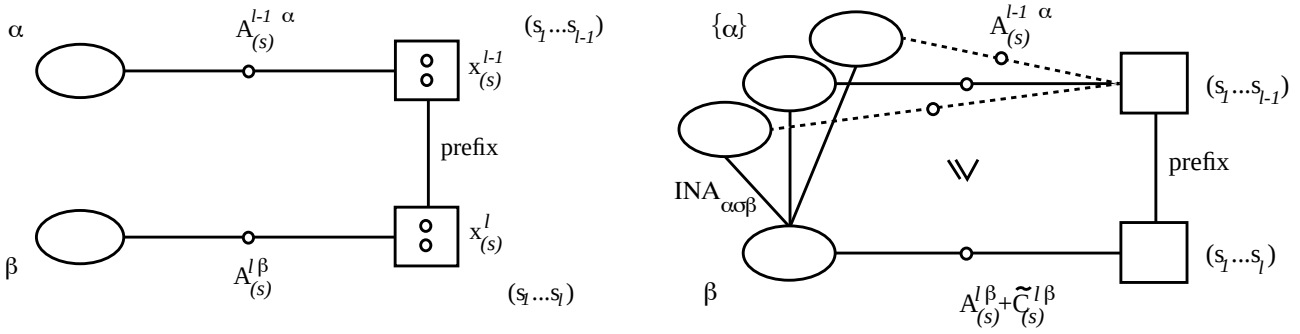

Figure 2: Aliveness variables in objective and constraint.

i.e.

$$E(C,\omega,x) = H_0\left(x^0\right) + \sum_{l=1}^{L}\sum_{(s)}\sum_{\alpha\beta} C^{l\,\beta}_{s_1\ldots s_l} C^{l-1\alpha}_{s_1\ldots s_l} \left(\prod_{k=1}^{l}\omega^{k}_{s_1\ldots s_l}\right)\left[H^{(\alpha s_l\beta)}\left(x^{l}_{s_1\ldots s_l}, x^{l-1}_{s_1\ldots s_{l-1}}, u^{(\alpha s_l\beta)}\right) - \mu^{(s_l\beta)}\omega^{l}_{(s)}\right] \quad (1)$$

and the constraints are:

$$\begin{gathered} C,\omega,P,\mathrm{INA} \in \{0,1\} \\ \omega^{l}_{(s_1\ldots s_l)} \le \sum_{\beta} C^{l\,\beta}_{s_1\ldots s_l} \le \prod_{k=1}^{l-1}\omega^{k}_{s_1\ldots s_l} \\ x_k = \sum_{(s)} P_{(s)k} x_{(s)} \quad\wedge\quad \sum_{(s)} P_{(s)k} = 1 \quad\wedge\quad \sum_{k} P_{(s)k} = \prod_{k=1}^{L}\omega^{k}_{s_1\ldots s_l} \le 1 \\ C^{l\,\beta}_{s_1\ldots s_l} \le \sum_{\alpha} \mathrm{INA}_{\alpha s_l\beta} C^{l-1\alpha}_{s_1\ldots s_{l-1}} \end{gathered} \quad (2)$$

Note that the constraint $\omega^{l}_{(s)} \le \sum_{\beta} C^{l\,\beta}_{(s)} \le \prod_{k=1}^{l-1}\omega^{k}_{(s)}$ has been used to simplify the objective in equation (1), by removing a product over C's from earlier levels. Note also that the **object→{preobject}** grammar rule could be restricted so that it couldn't delete preobjects (and only the preobject deletion rule could do so), e.g. $C^{l-1\alpha}\omega^{l-1} = 1 \Rightarrow \sum_{\beta} C^{l\beta}_{(s\sigma)} = \max_{\beta}\mathrm{INA}_{\alpha\sigma\beta}$. But the extra condition would later be eliminated along with the auxiliary variables of section below.

The mapping from such constrained optimization problems to neural nets is discussed in section 3.2. This particular problem formulation (1),(2) is difficult to work with because it has high polynomial degree (L+4 or more depending on H), making circuit implementations and global optimization difficult, and because there are several sets of variables indexed by the possible derivation paths $(s) = (s_1\ldots s_l)$, of which there are exponentially many as the maximum depth L increases. If the probability of preobject deletion is not very low, this leads to polynomial or even exponential discrepancies between the number of image points (which is the size of the input data to a recognition problem) and the number of possible intermediate terms which have to be represented to parse, explain or recognize a given scene. So the cost of storing an arbitrary configuration, let alone computing the optimal one, is prohibitive. Reformulating the constrained optimization problem to remove these obstacles is addressed in the next section.

3.0 Problem Reformulations

The cost and polynomial degree of the constrained optimization problem (1), (2) may be improved by repeatedly changing to a better set of variables. In this section we introduce the conditions that must be verified for a change of variables to produce an equivalent Boltzmann probability distribution and optimization problem.

Given a free energy objective, $F_1(x;T)$, depending on some set of variables x and on a fixed global temperature parameter T, and given a constraint predicate on x, $C_1(x)$, we can change variables to another set of variables y, free energy $F_2(y;T)$ and constraint predicate $C_2(y)$ by finding two change-of-variable predicates, $(\Delta_F(Y,X), \Delta_B(X,Y))$ (for the forwards and backwards directions respectively), usually expressed as conjunctions of equality and inequality constraints between supersets X and Y of x and y, such that:

$$\begin{gathered} C_1 \wedge \Delta_F \Leftrightarrow C_2 \wedge \Delta_B \qquad \text{and} \\ C_1 \wedge \Delta_F \wedge C_2 \wedge \Delta_B \Rightarrow F_1(x) - TS_1(X\backslash x) = F_2(y) - TS_2(Y\backslash y) - T\log J(x,y) \end{gathered} \quad (3)$$

The first condition says that the constraint space on X and Y is the same, whether it is described by a set of boolean constraints on x and a forward change of variables, or by a set of constraints on y and a backwards change of variables. The second condition says that in this common constraint space, the free energies differ only by entropy terms S_1 and S_2 which account for the logarithm of the volume of the allowed space of auxiliary variables $X\backslash x$ and $Y\backslash y$ (where \ is the set difference), and the logarithmic ratio of volume elements of any continuous variables among x and y. The entropy and Jacobian terms arise from the partition function in the statistical mechanics treatment of Boltzmann probability distributions, and may be ignored at zero temperature (T=0). For simplicity we'll ignore these terms and just verify the equality of objective functions E before and after the change of variables, but it is easy to reinstate the temperature-dependent terms.

$$\rightarrow \{\textbf{preobject}\left(l, \beta, (s_1 \ldots s_l), x^{l}_{s_1 \ldots s_l}\right) \Big| 1 \le s_l \le s \wedge C^{l\beta}_{s_1 \ldots s_l} = 1\}$$

$$E(C, x) = \sum_{s_l} \sum_{\beta} C^{\ l\ \beta}_{s_1 \ldots s_l} H^{(\alpha s_l \beta)}\left(x^{\ l}_{s_1 \ldots s_l}, x^{\ l-1}_{s_1 \ldots s_{l-1}}, u^{(\alpha s_l \beta)}\right)$$

$$\ni \ C^{\ l\ \beta}_{s_1 \ldots s_l} \le \mathrm{INA}_{\alpha s_l \beta} \ \wedge \ \sum_{\beta} C^{\ l\ \beta}_{s_1 \ldots s_l} \le 1$$

preobject $\left(l, \beta, (s_1 \ldots s_l), x^{l}_{s_1 \ldots s_l}\right)$ $\rightarrow$ **object** $\left(l, \beta, (s_1 \ldots s_l), x^{l}_{s_1 \ldots s_l}\right) \Big| \omega_{(s)} = 1$; $E = -\mu^{(s_l \beta)}$

$\rightarrow$ **nothing** $|\omega_{(s)} = 0$; $E = 0$ // ω is a choice param for one object

// So $E(\omega) = -\mu^{(s_l \beta)} \omega_{s_1 \ldots s_l}$

object $\left(L, \beta, (s_1 \ldots s_l), x_{s_1 \ldots s_L}\right)$ $\rightarrow$ **point** $\left(x^{\ L}_{s_1 \ldots s_L}\right)$; $E = 0$

{point $\left(x^{\ L}_{s_1 \ldots s_L}\right)$ **}** $\rightarrow$ **{imagepoint** $\left(x_k = \sum_{(s)} P_{(s)k} x^{L}_{(s)}\right)$ **};** $E = 0$

$$\ni \ \sum_{(s)} P_{(s)k} = 1 \ \wedge \ \sum_k P_{(s)k} = \prod_{l=1}^{L} \omega^{\ l}_{s_1 \ldots s_l} \le 1$$

// P is a permutation of $k_{\max}$ elements

The object-part recursion proceeds according to a fixed part-whole hierarchy between models (or types, or frames) α and β, if they are related by the constant matrix INA, and will result in a kind of semantic network we call "Frameville" [18][17]. The other variables may be explained as follows: l is the level number, bounded by L; α is the object type or "frame" index; (s) is a derivation path index consisting of a string of slot indices s_l; x is a set of analog parameters for an object, such as the position vector of its centroid; u is a set of constant analog parameters for a model, such as the expected displacement vector from the parent object's centroid to that of a child object; H is the penalty function (e.g. a quadratic) for deviations from the expected relations between parent and child parameters; C are the 0/1-valued choice parameters recording which frame β was chosen as the type of each preobject child of a given object; ω is the 0/1-valued choice parameter recording whether a preobject was deleted, or survived to become an object; and P is the random permutation that renumbers the level-L objects (now points) as "imagepoints" in an uninformative order. An example two-level derivation is shown below, in which transitions according to the level-1 object rule, the level-2 object rule (in two steps: part generation and point jitter), and the final point renumbering rule are shown:

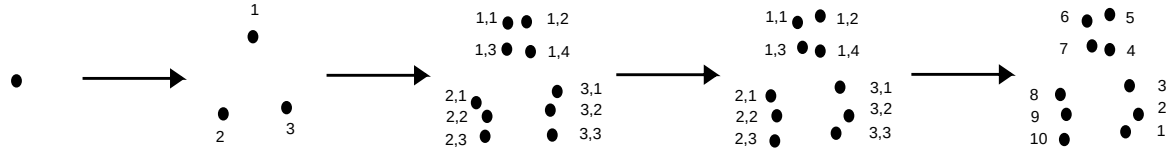

Figure 1: Generating a hierarchical object image

Because the model or "frame" indices α and β are term parameters in the grammar, the two mutually recursive object↔preobject rules are able to subsume many parameterized L-system rules of the form $\alpha(x) \rightarrow \{\beta(x)\}$, in which the allowed transitions are given by the INA array. (If several different rules start with the same $\alpha(x)$, we introduce intermediate models $\alpha_r(x)$ to select among them in two steps, $\alpha(x) \rightarrow \alpha_r(x) \rightarrow \{\beta(x)\}$. This is still encoded in INA, using its ability in this grammar to simulate ISA links.) In this way, the single six-rule grammar above is actually a grammar schema for any number of much larger grammars whose transitions can be tabulated by INA arrays. Considered as a graph on models or "frames", INA is not restricted to being a tree or a directed acyclic graph (DAG). We can compute the joint probability distribution on all the possible parameterized terms which could arise in a derivation of this grammar; it is another Boltzmann distribution [16]. The problem of finding the most probable derivation giving rise to a given set of level-L final terms is the constrained optimization problem of minimizing the resulting energy function. The objective may be straightforwardly calculated as:

$$E(C, \omega, x) = H_0\left(x^0\right) + \sum_{\alpha_1 s_1} C^{1\alpha_1}_{s_1}\left(H^{(\mathrm{scenes}_1 \alpha_1)}\left(x^1_{s_1}, x^0, u^{(\mathrm{scenes}_1 \alpha_1)}\right) - \mu^{(s_1 \alpha_1)} \omega^1_{s_1}\right.$$

$$+ \omega^1_{s_1} \sum_{\alpha_2 s_2} C^{2\alpha_2}_{s_2}\left(H^{(\alpha_1 s_2 \alpha_2)}\left(x^2_{s_1 s_2}, x^1_{s_1}, u^{(\alpha_1 s_1 \alpha_2)}\right) - \mu^{(s_2 \alpha_2)} \omega^2_{s_2}\right.$$

$$\left. + \ldots + \omega^{L-1}_{s_1 \ldots s_{L-1}} \sum_{\alpha_L s_L} C^{L\alpha_L}_{s_1 \ldots s_L}\left(H^{(\alpha_{L-1} s_L \alpha_L)}\left(x^{\ L}_{s_1 \ldots s_L}, x^{\ L-1}_{s_1 \ldots s_{L-1}}, u^{(\alpha_{L-1} s_L \alpha_L)}\right) - \mu^{(s_L \alpha_L)} \omega^L_{s_1 \ldots s_L}\right) \ldots\right)\right)$$

2.0 The Grammar

Our basic modeling tool is neither a neural net nor a symbol processor, but rather a type of stochastic grammar which permits quantitative, probabilistic modeling as well as considerable expressive power. The grammar is "parameterized" because each term carries numerical parameters whose values are determined when the right rule fires. Each rule has a conditional probability distribution on these parameters which could be any Boltzmann distribution; such a distribution by itself is an expressive language for many domains. The presence of multiple rules serves to integrate qualitatively different processes into one model.

As is common in high-level languages, we first design for appropriate expressiveness (now including learning and pattern recognition) and then worry about program transformations which can obtain high performance on a useful class of examples. Here the program transformations include different ways of translating inference problems on grammars to constrained optimization problems and thence to neural architectures which seek good local minima at low cost, as well as transformations within each of these stages. Such transformations [16][12][13] can be much more quantitative than conventional program transformations, and introduce great flexibility in the neural "wiring diagrams" that correspond to a given grammar model. They may also include outright approximations, such as our use of deterministic annealing in place of global optimization. This transformational approach gives us new methods with which to explore the inevitable trade-offs between expressiveness and efficiency.

In a parameterized L-system grammar [23], rules can fire in parallel and terms can carry parameters which are altered by the rules. Our stochastic grammar rules will be of the form:

term(*params*)**:** l-condition(*params*)→ {**term1**(*params1*), **term2**(*params2*), ... | r-condition(*choice-params*)}; // comments

$E = E$ ({*paramsi*}, *choice-params*; *params*) // comments

ə constraints({*paramsi*}) // comments

In this rule schema, E is the energy function in a Boltzmann probability distribution on the parameters {*paramsi*} of the new terms **termi**, given the fixed parent **term** and its fixed *params*. The constraint introduction symbol "ə " is read "such that", and the constraints are predicates (usually conjunctions of equalities or inequalities) which restrict the space of allowed parameter values for the new terms. The optional left and right conditions are predicates, depending on the parent and choice parameters respectively, which must be satisfied for the rule to fire; they are introduced by symbols ":" and "|" which can be read as "for which" and "such that". The optional choice parameters are used to parameterize all the alternative sets of terms, {**termi**}, which could result from the same left hand side, to determine their relative probabilities in the relevant Boltzmann distributions, and to record numerically the choice made. The set of new terms could have just one element, in which case we'll omit the braces, and it is unordered except for any implicit order encoded in the parameters. This form of rule is context-free, but we will also use one context-sensitive rule in which the left hand side is a set of terms rather than a single term.

We now introduce a particular grammar for generating images consisting of sparse point-like features that represent hierarchical objects. A "scene" generates a single scene "object", which recursively generates other "objects" each of which can be randomly deleted, through L generations of a part-whole hierarchy. A generation-L object generates a "point", and the context-sensitive final rule randomly relabels all the points. The "object" terms will be of the form **object**(<level>, <type>, <deriv-trace>, <instance-params>), where <level> is an integer level number l, constrained to increase with each rule firing and to not exceed a maximal derivation depth L; <type> is the object type or model number α; <deriv-trace> is a record of the path to this node in the compositional (part-whole) hierarchy, in the form of a unique sequence $(s_1 s_2 \ldots s_l)$ of slot or sibling numbers; and <instance-params> are possibly analog parameters specifying, for example, the spatial coordinates and orientation of the object.

Each object has an ordered set of slots to be filled by its children in the compositional hierarchy. The children are pre-objects which, if they are not randomly deleted, survive to become objects. The child of an object of type α, whose path index is $(s_1 s_2 \ldots s_l)$, must choose some type β; the choice is constrained by the constant 0/1 array of allowed component types $\mathrm{INA}_{\alpha s_l \beta}$, and recorded in the 0/1-valued choice variables $C^{l\beta}_{s_1 \ldots s_l}$. In detail, the grammar is:

scene → **object**(l=*0*, α="scene", ( ), x^0); $E = H^0(x_0)$

object$\left(l-1, \alpha, (s_1 \ldots s_{l-1}), x^{l-1}_{s_1 \ldots s_{l-1}}\right)$ **:** $l < L$

To appear in: *Connectionist Symbolic Integration*, eds. R. Sun and F. Alexandre, Lawrence Erlbaum Associates.

Symbolic Neural Networks Derived from Stochastic Grammar Domain Models

Eric Mjolsness
Institute for Neural Computation, and
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

Abstract

Starting with a statistical domain model in the form of a stochastic grammar, one can derive neural network architectures with some of the expressive power of a semantic network and also some of the pattern recognition and learning capabilities of more conventional neural networks. For example in this paper a version of the "Frameville" architecture, and in particular its objective function and constraints, is derived from a stochastic grammar schema. Possible optimization dynamics for this architecture, and relationships to other recent architectures such as Bayesian networks and variable-binding networks, are also discussed.

1.0 Introduction

This paper outlines a statistical approach to unifying certain symbolic and neural net architectures, by deriving them from a stochastic domain model with sufficient structure. The domain model is a stochastic L-system grammar, whose rules for generating objects and their parts each include a Boltzmann probability distribution. Using such a domain model in high-level vision, it is possible to formulate object recognition and visual learning problems as constrained optimization problems [16] of a restricted form which can be locally optimized by suitable neural network architectures. In ths way, one can semi-automatically produce neural nets for object recognition with some of the representational properties of semantic nets.

In technical content, this paper generalizes the method in several ways: it includes a fairly general recursive grammar in which derivations may have a large but bounded depth; it allows the compositional hierarchy of object and part models or types to be a general graph; it subsumes a specialization hierarchy of object models within the compositional hierarchy; it adds learning as another constrained optimization problem; it introduces a general way to reformulate the optimization problems by "changing variables". However, many restrictions on the expressiveness and functionality of the combined architecture remain. The most important issue not addressed in this paper is the neural network dynamics by which the nonlinear global optimization problem is to be solved, for large systems. There has been substantial progress on this subject for related, less difficult combinatorial optimization problems [9][20][26] and their application to pattern recognition and learning in high-level vision [6][5] derived from somewhat simpler stochastic grammars [16].

Deriving the unified architecture from a statistical model has several advantages, including the integration of suitable "distance" or "similarity" measures for pattern recognition and learning on structured objects, and the specification of a broad class of domain models for which the symbolic neural network architecture is applicable. In Section 2 we will describe the grammar domain model and how it incorporates compositional and specialization hierarchies (as present in semantic nets) within a parameterized grammar schema. In Section 3 we summarize some of the technical manipulations by which probabilistic parsing and grammatical inference problems can be cast as constrained optimization problems with various cost characteristics, including the derivation of a semantic net-like formulation with instantiation links, and solved by neural networks. Section 4 compares the present approach to symbolic/neural net unification with several related architectures, and concludes.