Useful ideas for exploiting time to engineer representations [*]

Richard Rohwer
Dept. of Computer Science and Applied Mathematics
Aston University, Birmingham B4 7ET, UK
rohwerrj@uk.ac.aston.cs

25 September 1992

Shastri and Ajjanagadde have found an interesting way to exploit the representational potential of time in neural network models. In most 'neural software engineering', a correspondence is defined between some of the state vectors of the model and interpretations in an application domain. The representational power of a state is limited by its dimension; *eg.*, a network of N binary-valued nodes can represent at most 2^N different things. But without allocating any further hardware resources, that representational power can be increased to 2^{NT} by interpreting length-T temporal sequences of states instead of individual states. It's a space-time tradeoff: It takes T times longer to represent something this way, but 2^T times as many things are representable.

The authors have found a situation in which this trade-off is an impressively good deal. It is important to have the power to represent a great variety of variable bindings, but most will never *actually* get represented in practice, and most of those that do will not need to be represented for very long. So it is better to spend some time rebuilding the representational setting each time a binding needs to be represented than to keep lots of spare representational capacity on tap.

The space-time tradeoff in this system is partly illusory, because its dynamics is order $T-1$ in the state variables, where T is the number of phases in a fundamental period. This is because maintenance of synchrony requires connections with time-delay $T-1$ between the ρ-btu nodes representing corresponding parts of rule-related predicates. Consequently, so far as the dynamics is concerned, a 'state' has $N(T-1)$ components.

[*]Comment for *Behavioral and Brain Sciences* target article "From Simple Associations to Systematic Reasoning: A Connectionist representation of rules, variables, and dynamic bindings using temporal synchrony", by Lokendra Shastri and Venkat Ajjanagadde.

Whether temporal synchrony is implemented with simple delay lines or the elaborate mechanism in section 7.3, a buffer of size $N(T-1)$ has to be directly or indirectly implemented in order for the system to run. These extra degrees of freedom can be thought of as implemented at a sub-cellular level. Computer simulations have to dedicate memory to them.

Although temporal coincidence plays a key role in this system, the oscillations seem inessential to its operation. What matters is that fact predicates 'observe' whether their arguments fire synchronously with any constants at least once during a reasoning episode, and that variables linked by rules eventually fire at the same time as any constant to which they may be bound. Periodic reiteration of these coincidences seems a waste of time. The only important role of the oscillations is in keeping variables linked by rules synchronised with each other. That way a constant synchronised with one is synchronised with all. The synchronisation among rule-related variables would be maintained by instantaneous propagation of activations, if only that were possible. Instead it is achieved (eventually) by delaying propagation for nearly one basic oscillation period, or by more elaborate mechanisms which require at least one cycle to take effect. Perhaps there is a cheaper way.

This system's elegant distribution of representations over time is not matched by an elegant distribution of representations over nodes. Grandmother-cell (or cell cluster) representations of constants and variables are used throughout. This may be just as well for expository purposes, but greater efficiency and potentially interesting properties may arise from more fully distributed representations. A set of C constants, for example, can be represented as patterns distributed over $O(\log C)$ nodes. (A more sparse representation using $\alpha \log C$ nodes, with $\alpha \gg 1$ but nevertheless $\alpha \log C \ll C$, might have more useful properties.) Smolensky, Dolan, and others have developed 'tensor product' binding methods which use distributed representations of constants and variables [?]. Unfortunately, these methods require $(\alpha \log C)(\alpha \log V)$ nodes to represent bindings among C constants and V variables. C and V refer to *all* constants and variables, not just those used in an episode of reasoning. It seems feasible, however, to distribute the tensor product over time, using a mixture of the tensor product binding and phase binding approaches [?]. This offers the combined advantages of each system. The total number of nodes required to represent the constants and variables is reduced from the grandmother-cell system's $O(C+V)$ to $O(\log C + \log V)$. No extra nodes are needed to represent the tensor product, but some extra time steps are needed, as many as there are bindings in the episode of reasoning. In addition to providing increased efficiency, the distributed representations might give such a system interesting generalisation properties found in the more popular neural network models.