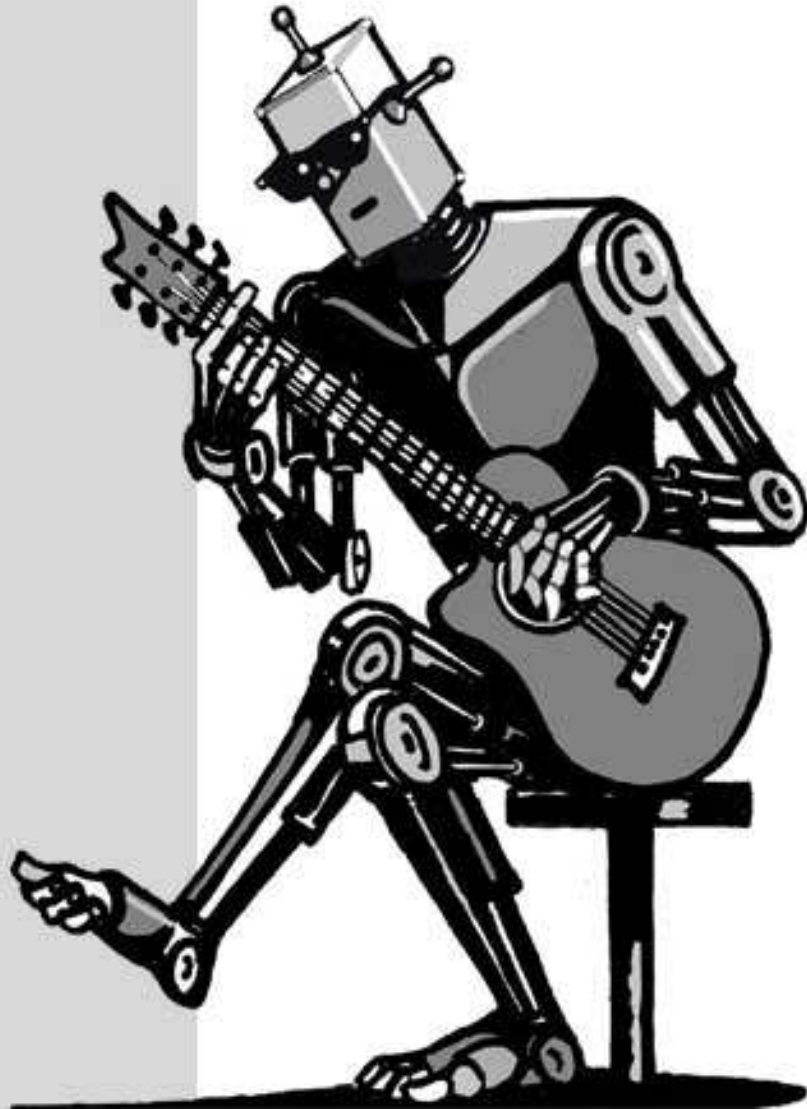




Learning how to Learn Learning Algorithms: Recursive Self-Improvement

Jürgen Schmidhuber
The Swiss AI Lab IDSIA
Univ. Lugano & SUPSI
<http://www.idsia.ch/~juergen>

NNAISENSE

Jürgen Schmidhuber
You_again Shmidhoobuh

“True” Learning to
Learn (L2L) is **not** just
transfer learning!

Even a simple
feedforward NN can
transfer-learn to learn
new images faster
through pre-training
on other image sets

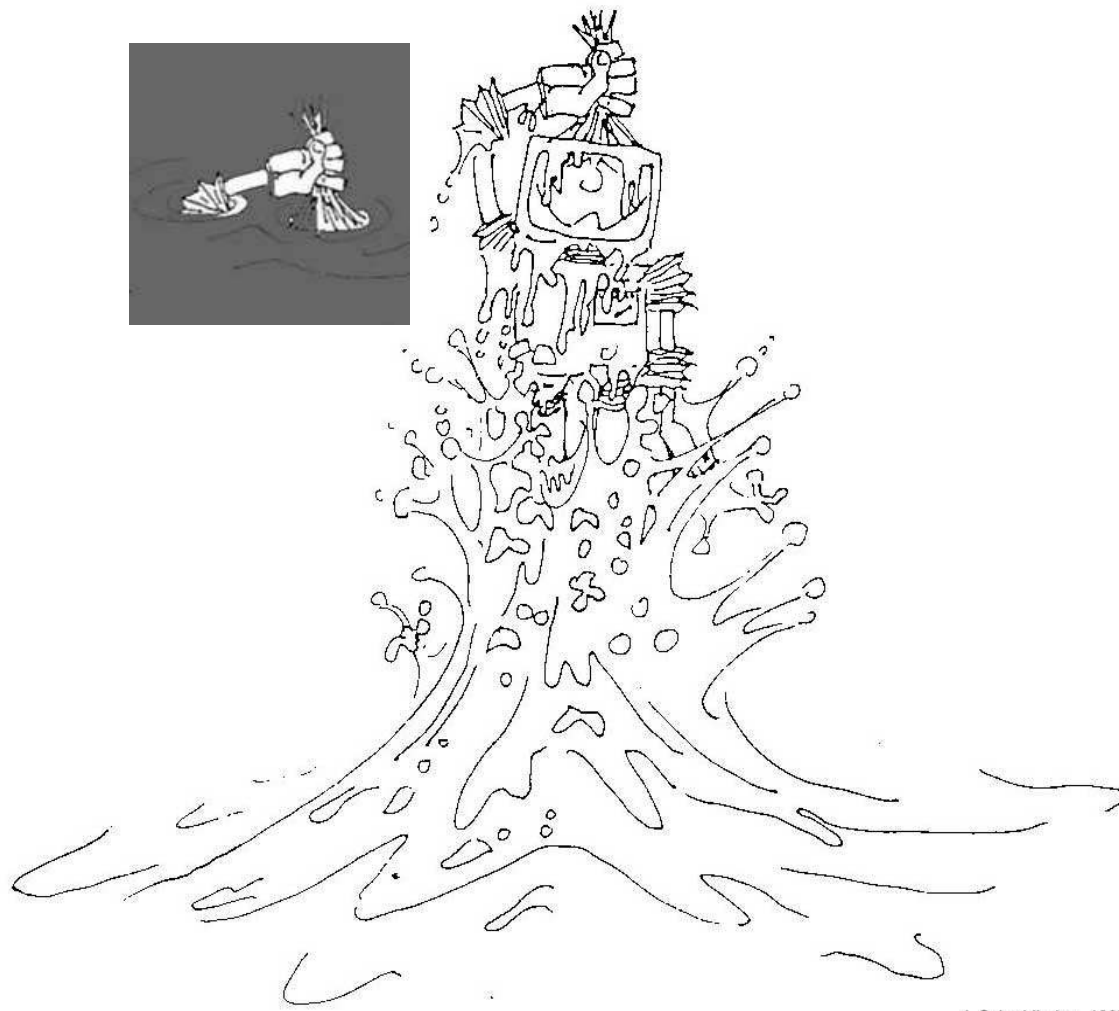
True L2L is **not** just
about learning to
adjust a few hyper-
parameters such as
mutation rates in
evolution strategies
(e.g., Rechenberg &
Schwefel, 1960s)

Radical L2L is about encoding the initial learning algorithm in a universal language (e.g., on an RNN), with primitives that allow to modify the code itself in arbitrary computable fashion

Then surround this self-referential, self-modifying code by a recursive framework that ensures that only “useful” self-modifications are executed or survive (RSI)

J. Good (1965): informal
remarks on an intelligence
explosion through recursive
self-improvement (RSI) for
super-intelligences

My concrete
algorithms for RSI:
1987, 93, 94, 2003

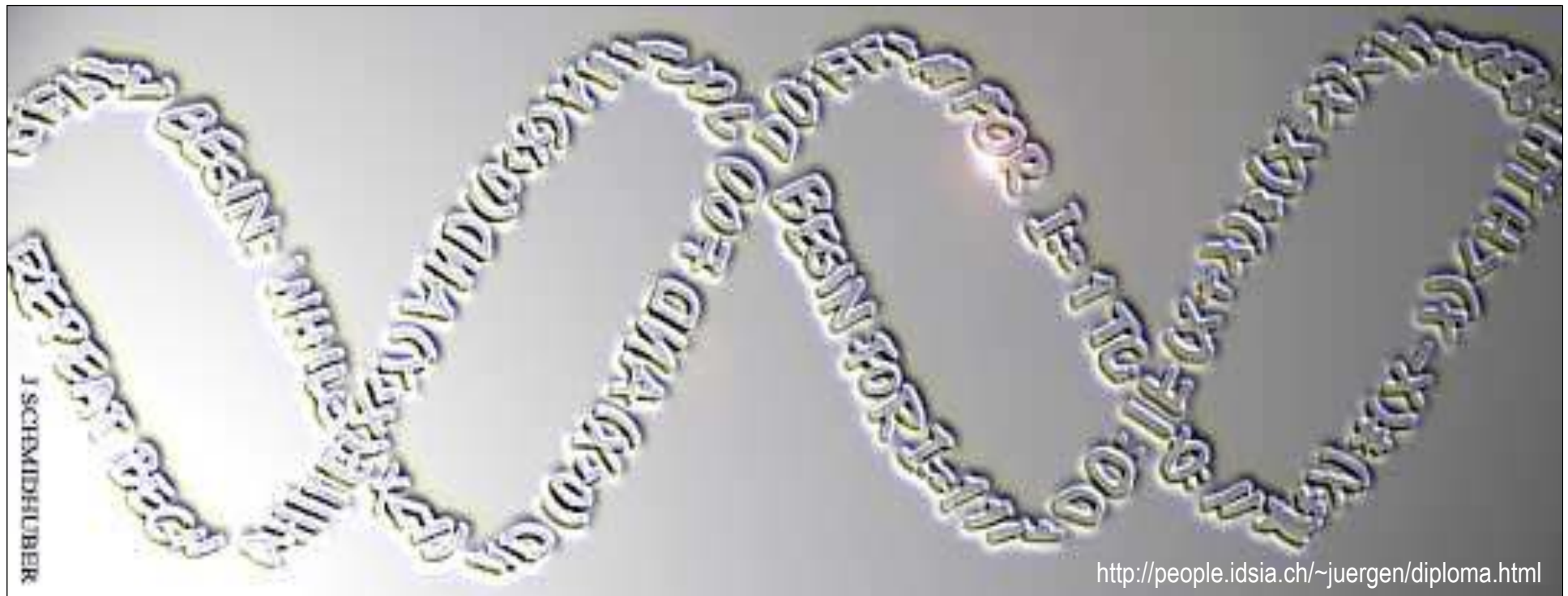


My diploma thesis (1987):
first concrete design of
recursively self-improving AI

<http://people.idsia.ch/~juergen/metalearner.html>

R-learn & improve learning
algorithm itself, and also the
meta-learning algorithm, etc...

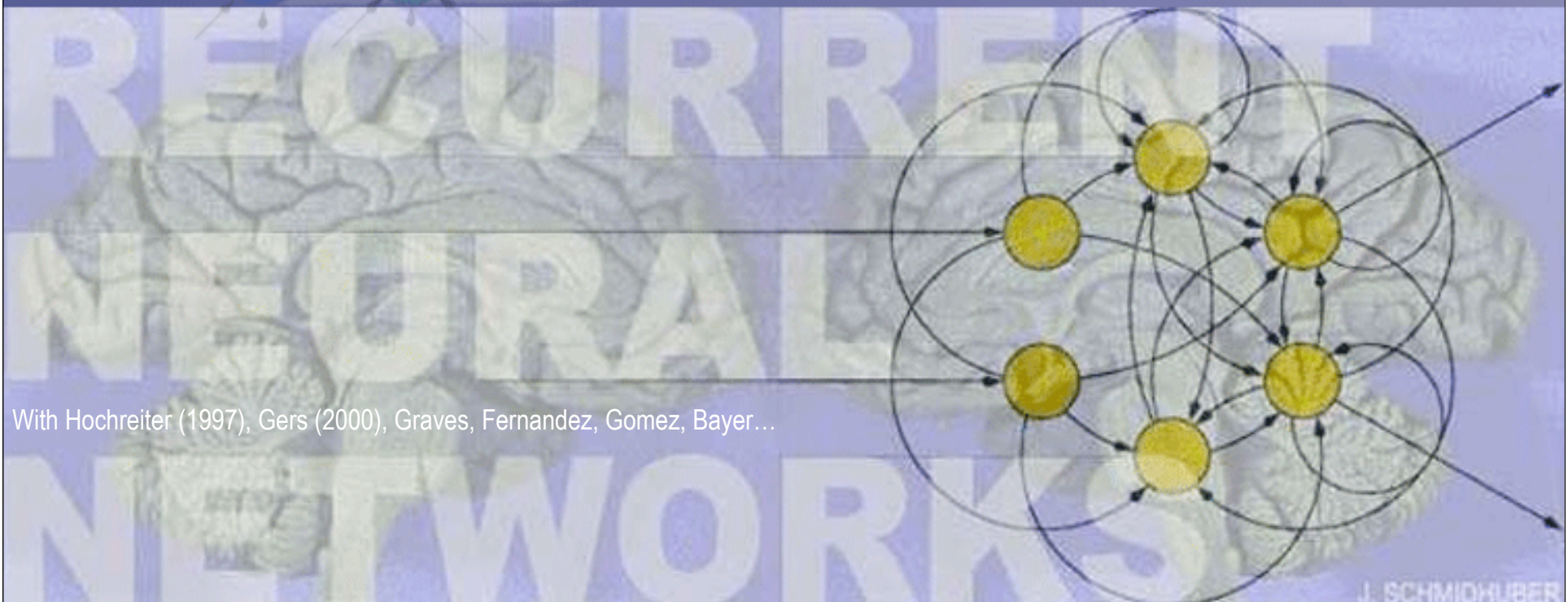
J. Schmidhuber, 1987



Genetic Programming recursively applied to itself, to obtain Meta-GP and Meta-Meta-GP etc: J. Schmidhuber (1987). Evolutionary principles in self-referential learning. On learning how to learn: The meta-meta-... hook. Diploma thesis, TU Munich

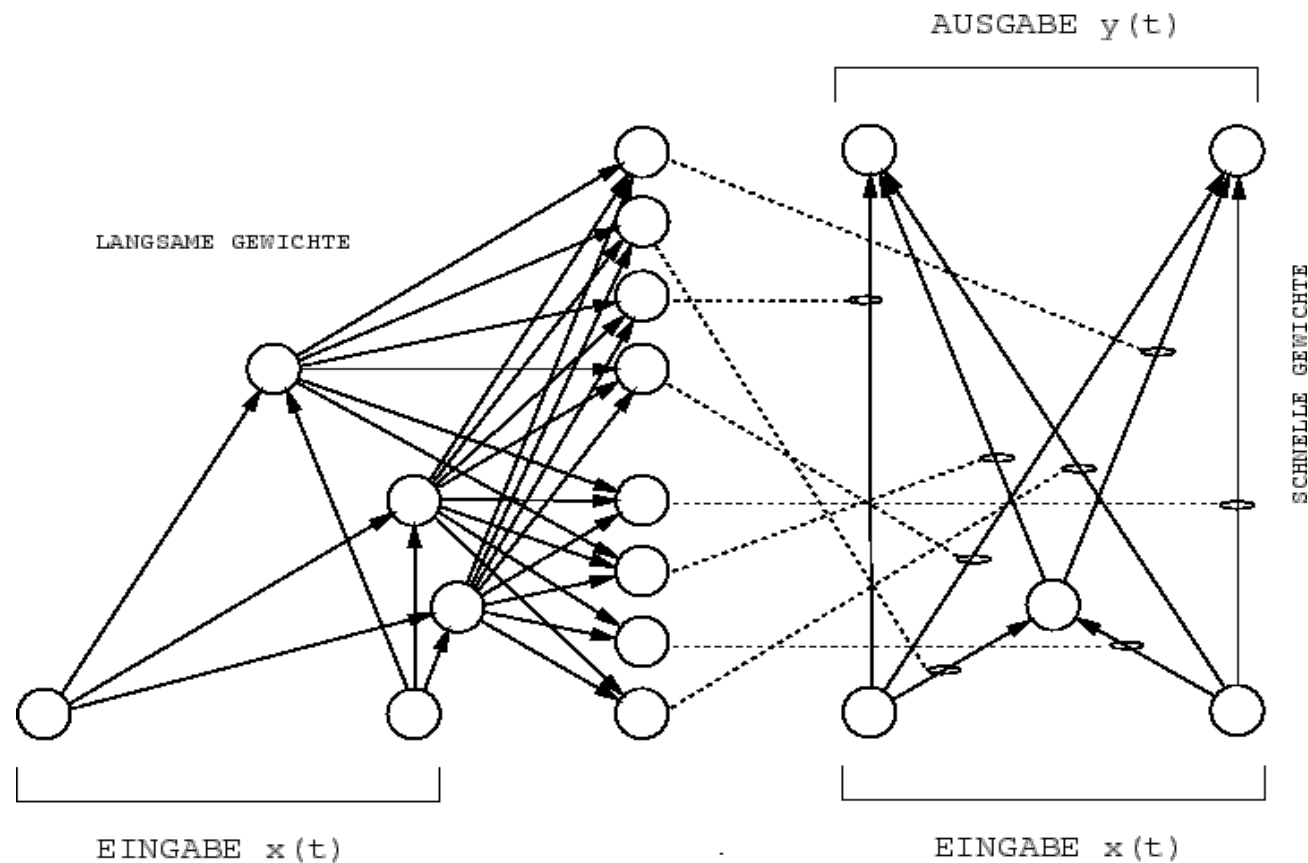
LONG SHORT-TERM MEMORY

1997-2009. Since 2015 on your phone! Google, Microsoft, IBM, Apple, all use LSTM now

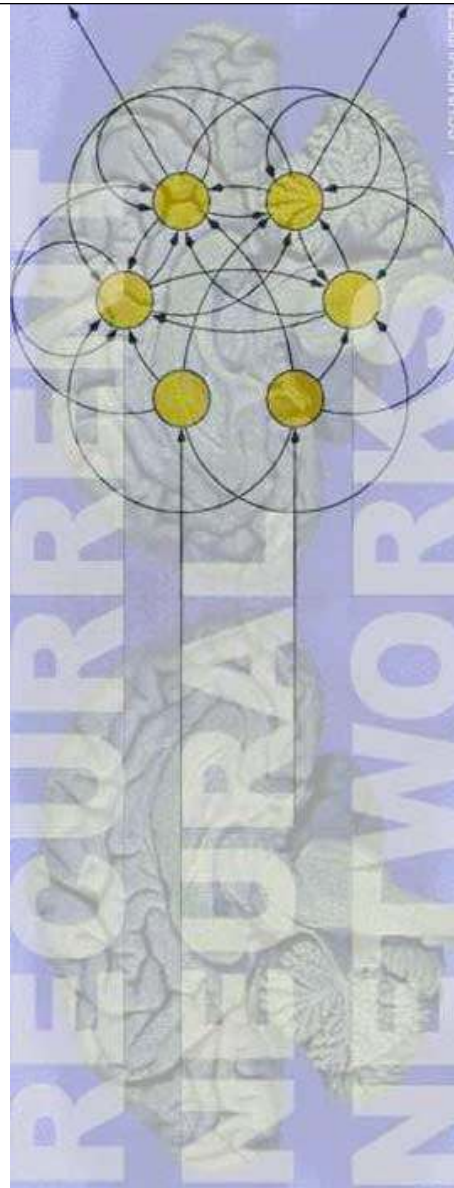


With Hochreiter (1997), Gers (2000), Graves, Fernandez, Gomez, Bayer...

Separation of Storage and Control for NNs: [End-to-End Differentiable Fast Weights](#) (Schmidhuber, 1992) extending v.d. Malsburg's non-differentiable dynamic links (1981)



1993: More elegant
Hebb-inspired
addressing to go
from $(\#hidden)$ to
 $(\#hidden)^2$ temporal
variables: gradient-
based RNN **learns**
to control internal
end-to-end
differentiable
spotlights of
attention for fast
differentiable
memory rewrites –
again **fast weights**

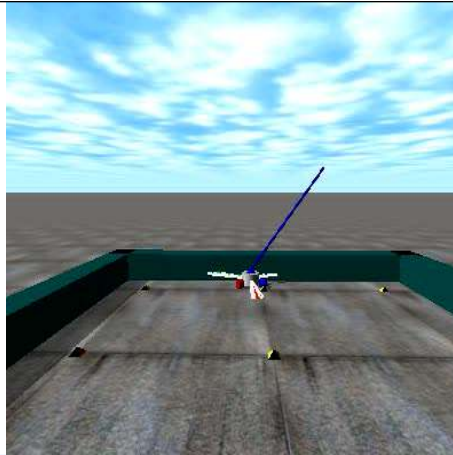


Schmidhuber,
ICANN 1993:

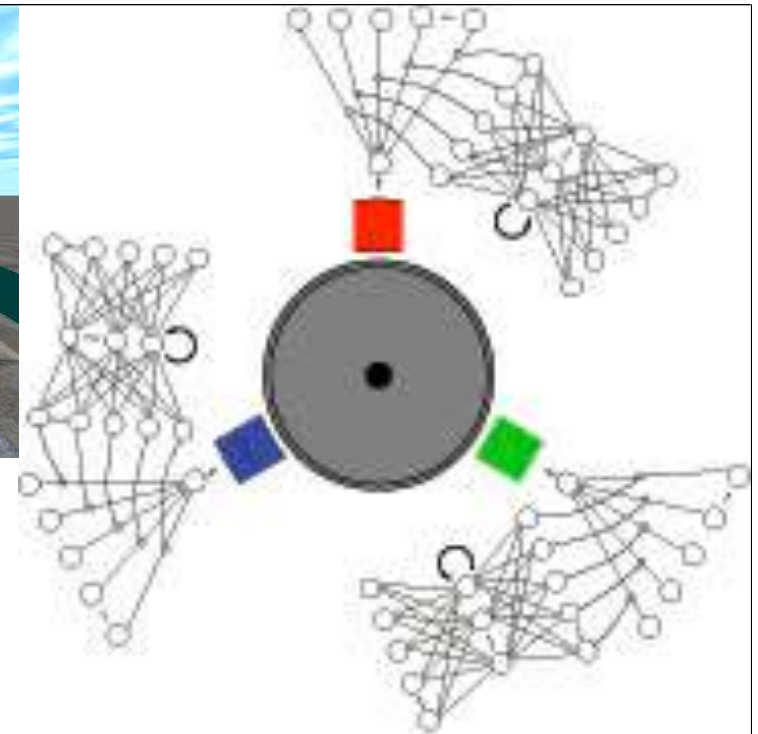
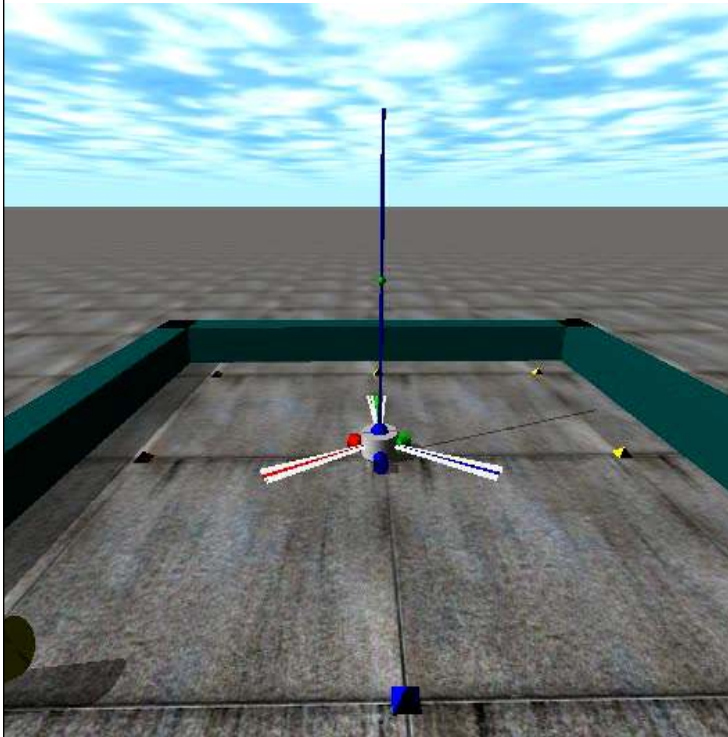
Reducing the ratio
between learning
complexity and
number of time-
varying variables in
fully recurrent nets.

Similar to NIPS
2016 paper by
Ba, Hinton, Mnih,
Leibo, Ionesco

2005:
Reinforcement-
Learning or
Evolving RNNs
with Fast Weights

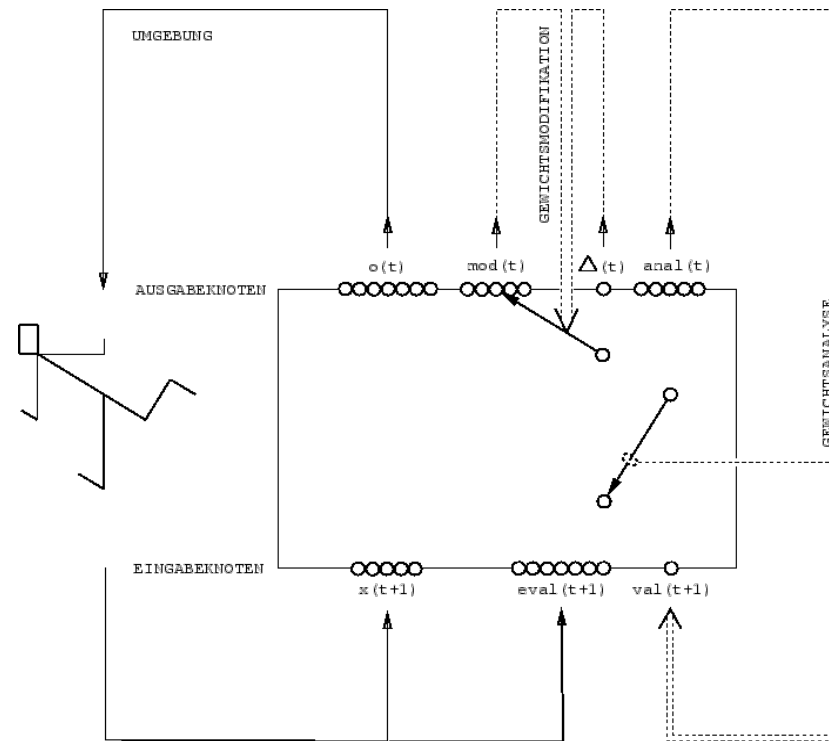
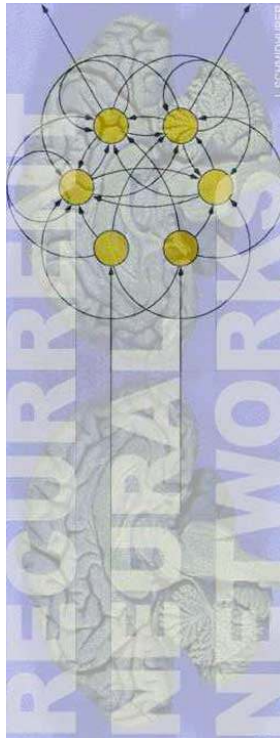


Robot learns to
balance 1 or 2 poles
through 3D joint



Gomez & Schmidhuber:
Co-evolving recurrent
neurons **learn deep**
memory POMDPs.
GECCO 2005

<http://www.idsia.ch/~juergen/evolution.html>



1993: Gradient-based meta-RNNs that can learn to run their own weight change algorithm:
J. Schmidhuber.
A self-referential weight matrix.
ICANN 1993

This was before LSTM. In 2001, however, Sepp Hochreiter taught a meta-LSTM to learn a learning algorithm for quadratic functions that was faster than backprop

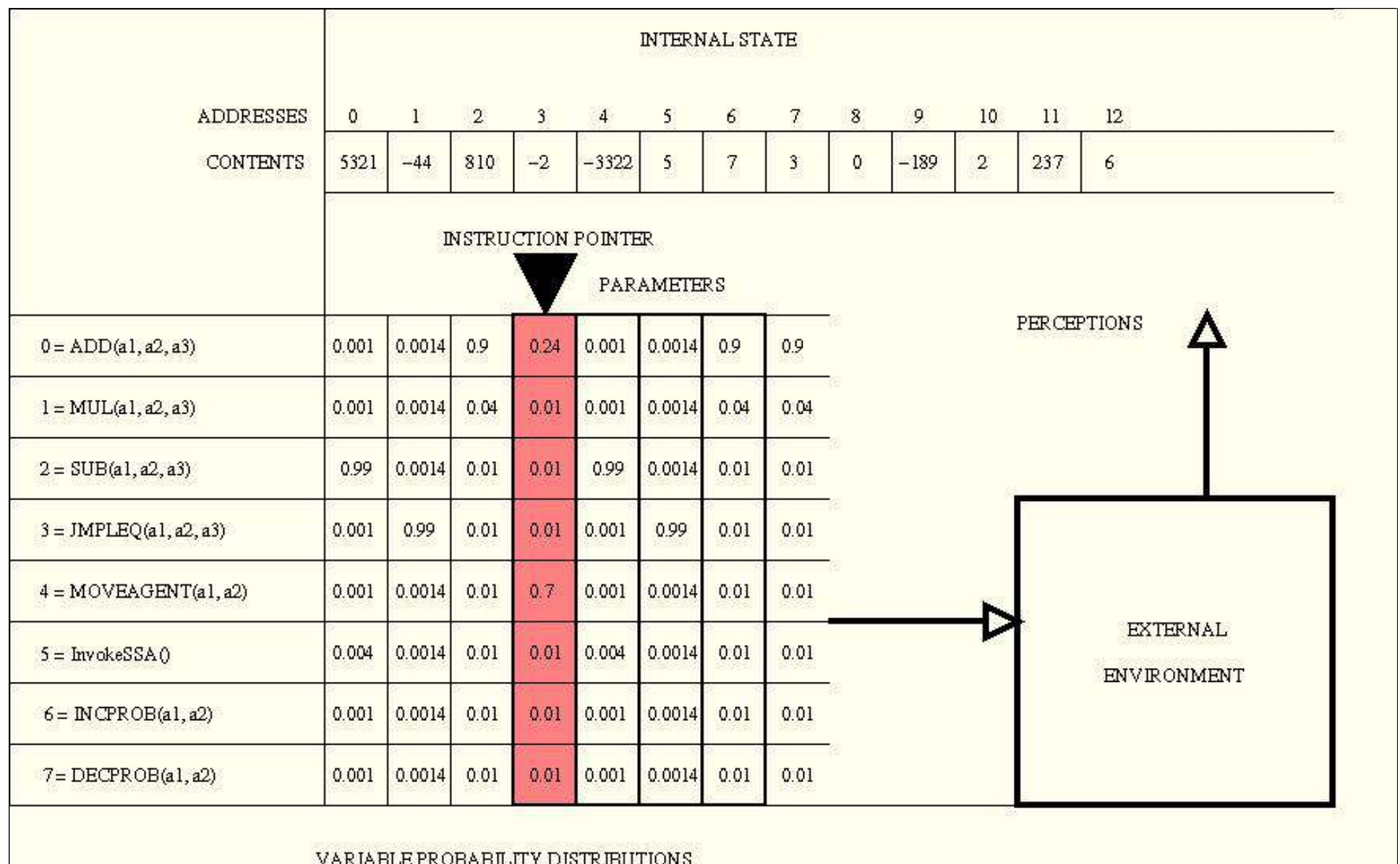
Success-story algorithm (SSA) for self-modifying code (since 1994)

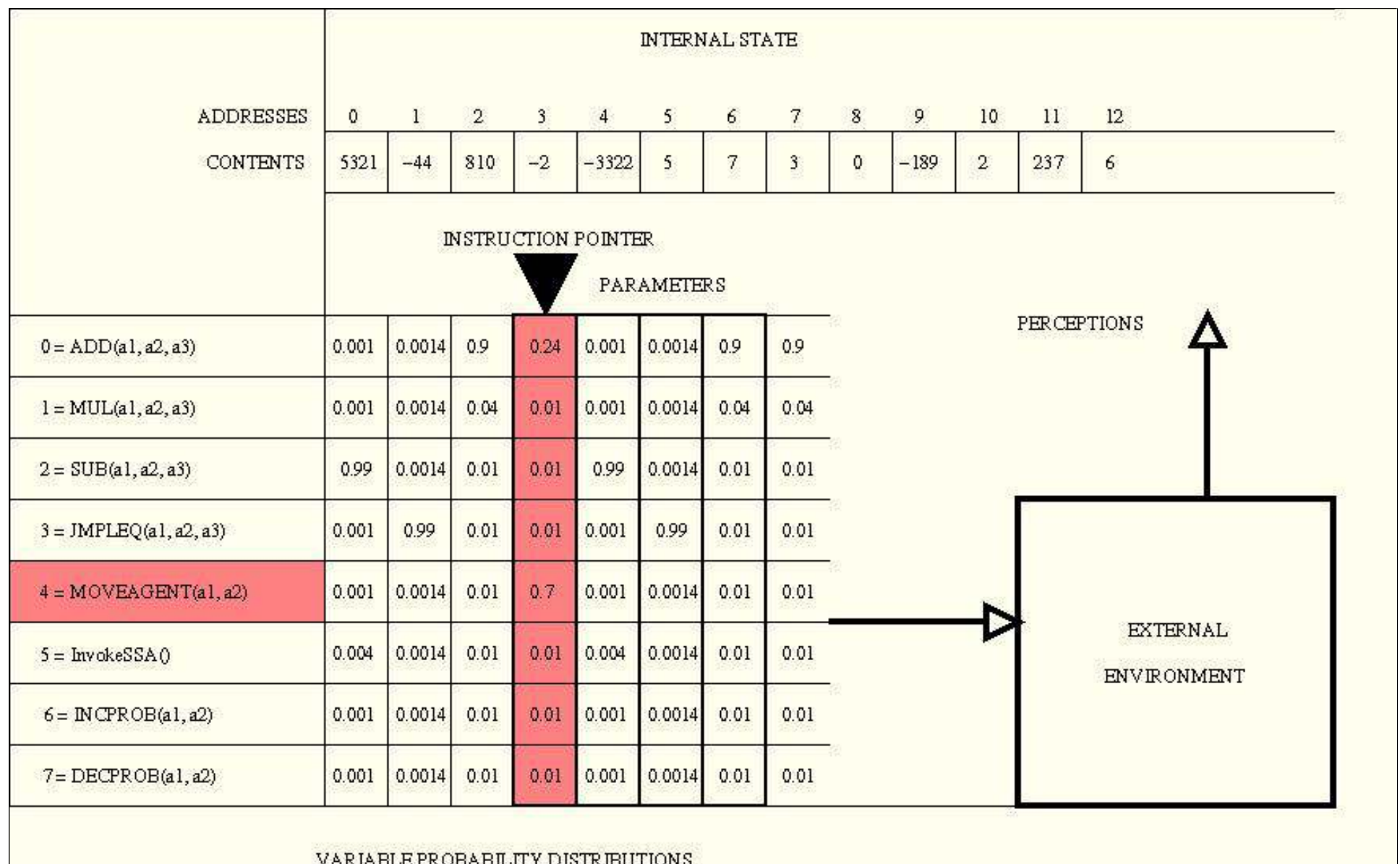
E.g., Schmidhuber,
Zhao, Wiering: MLJ
28:105-130, 1997

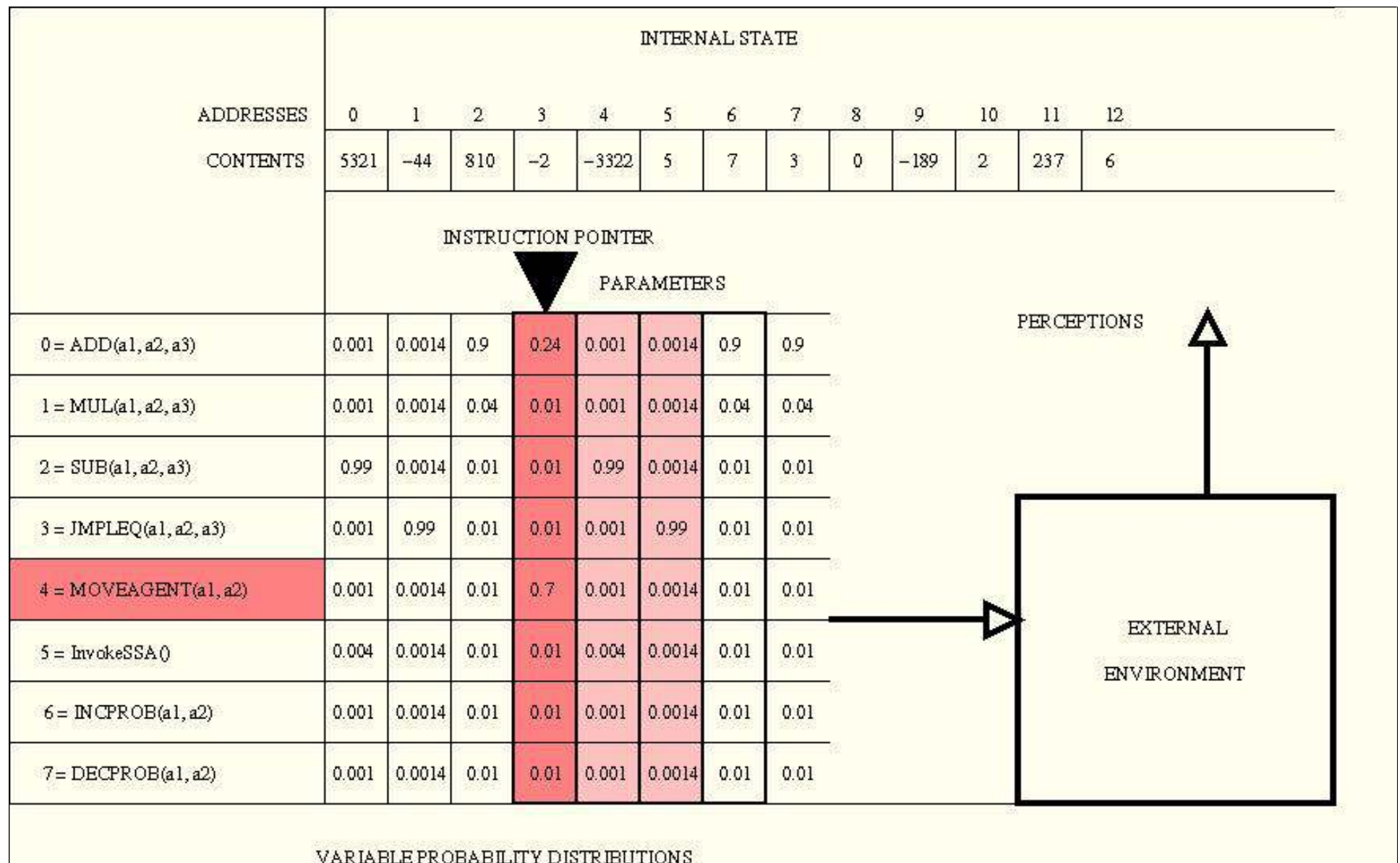
$R(t)$: Reward until time t . Stack of past check points $v_1 v_2 v_3 \dots$ with self-mods in between. SSA undoes selfmods after v_i that are not followed by long-term reward acceleration up until t (now):

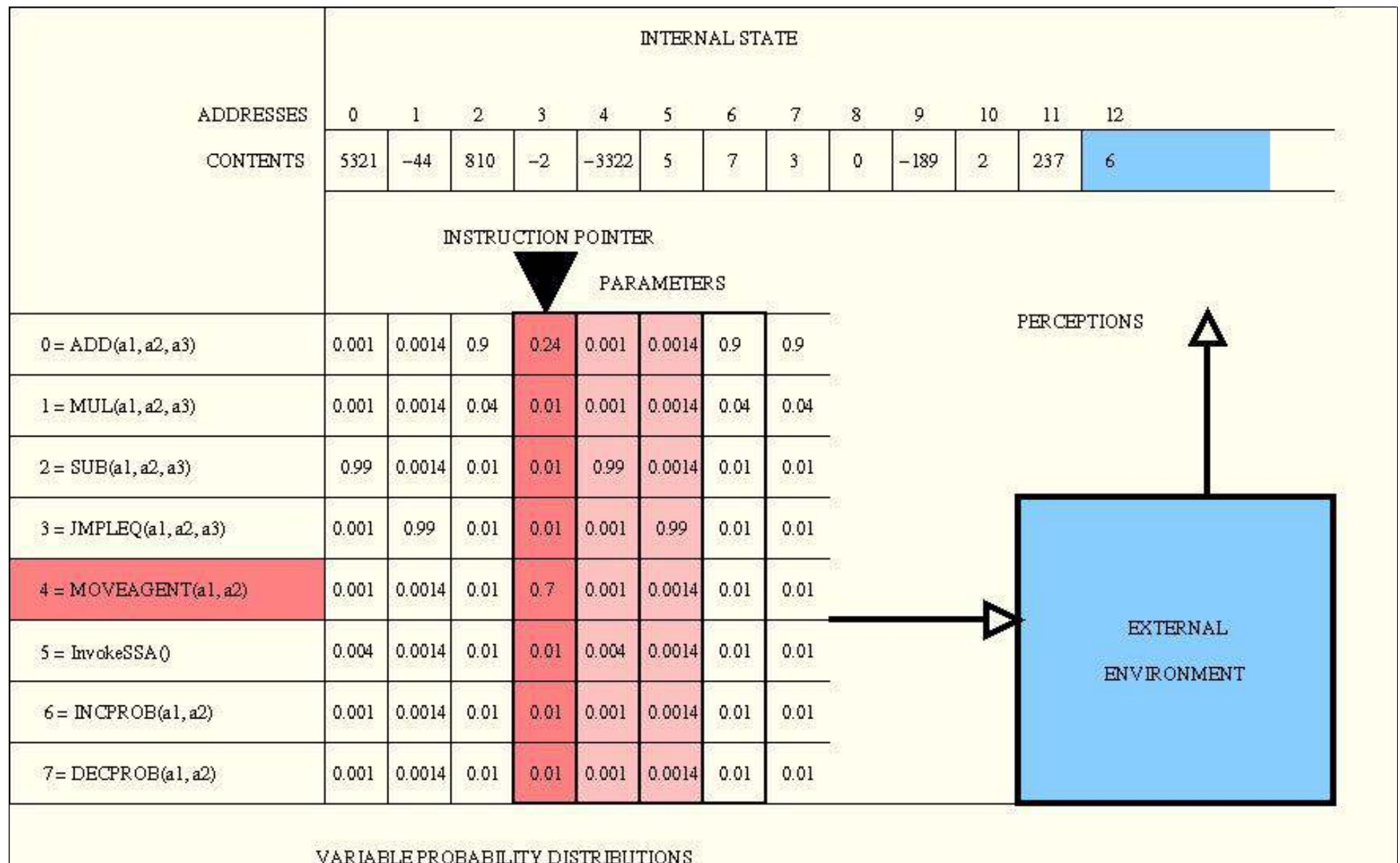


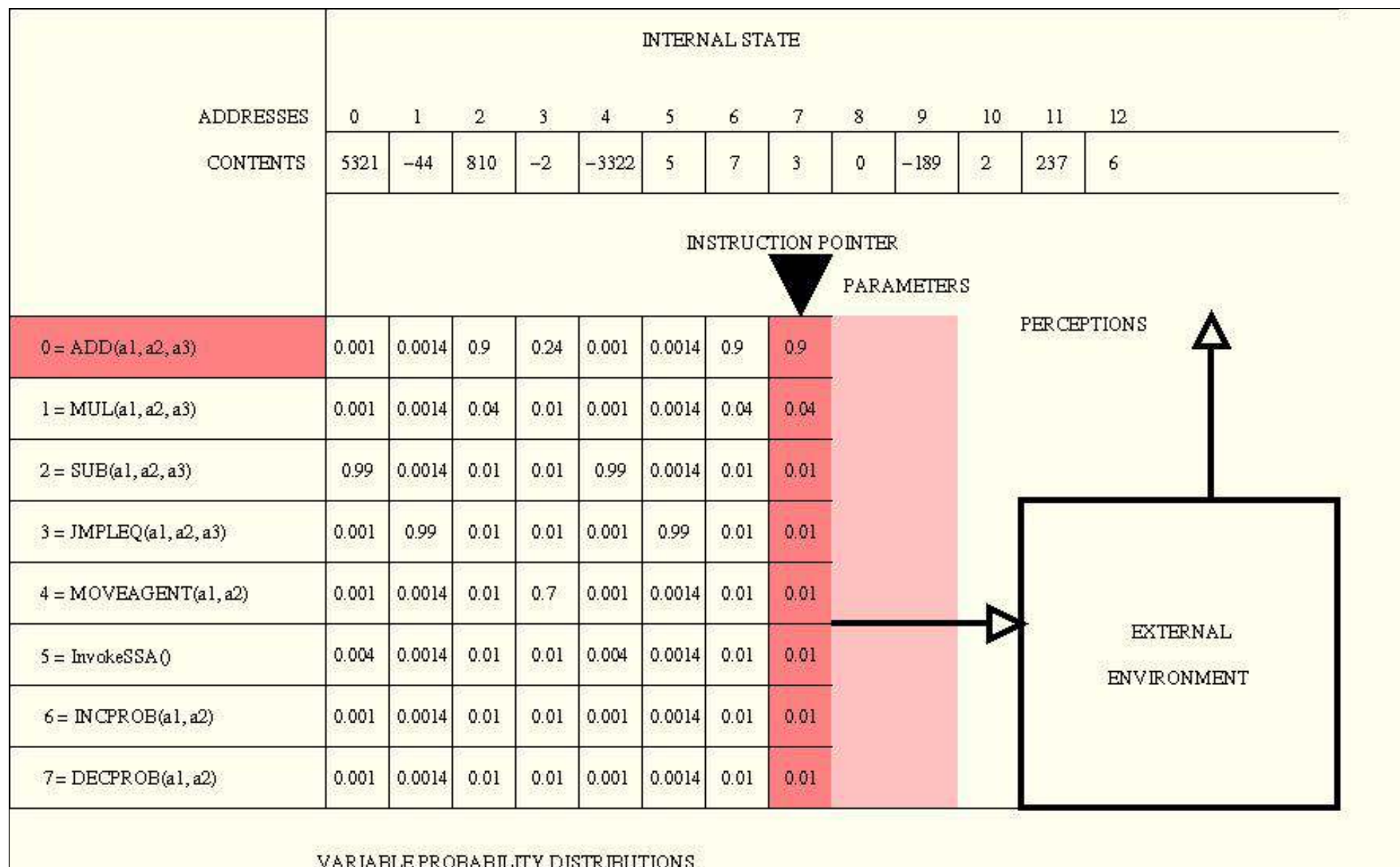
$$R(t)/t < [R(t)-R(v_1)] / (t-v_1) < [R(t)-R(v_2)] / (t-v_2) < \dots$$

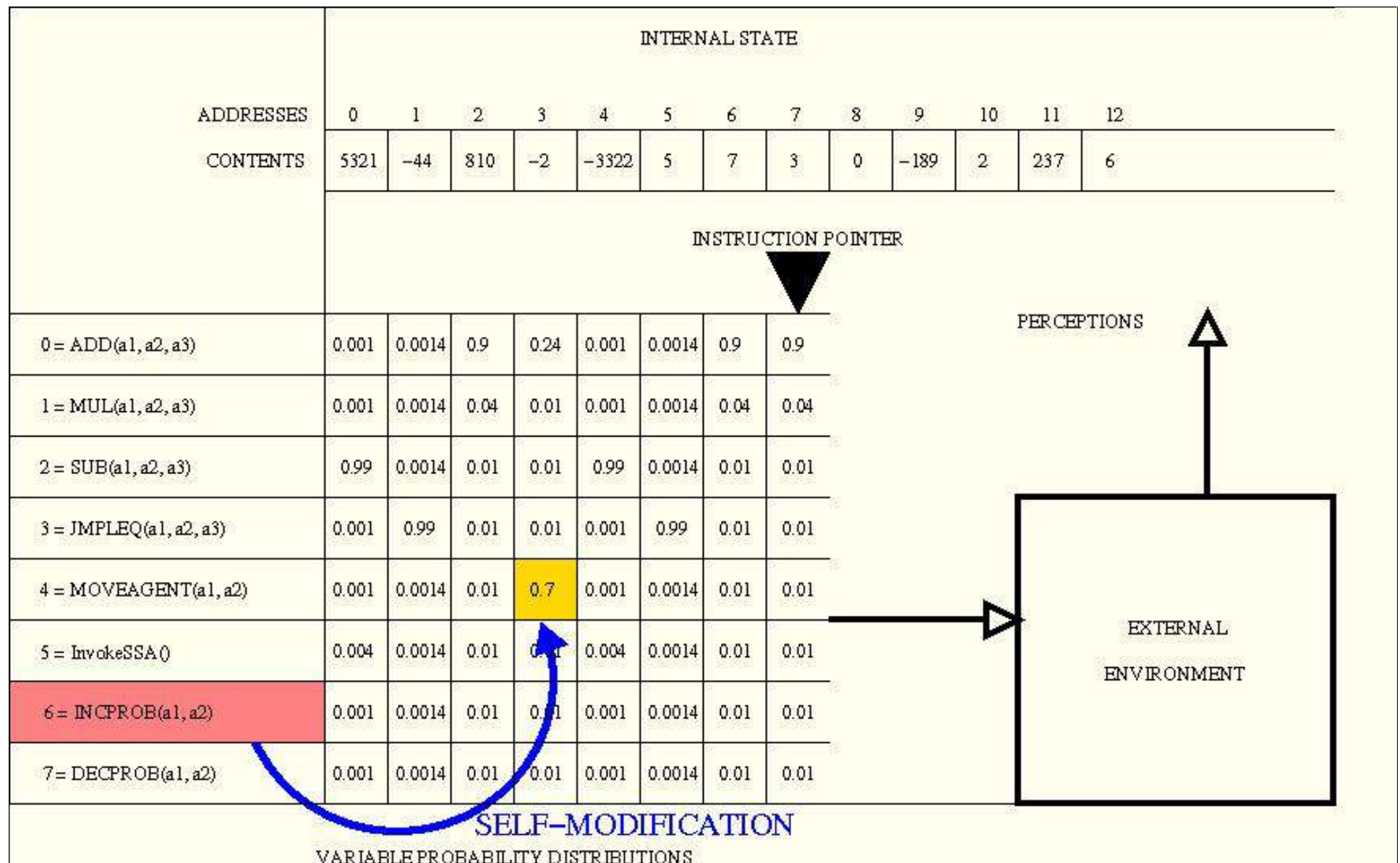


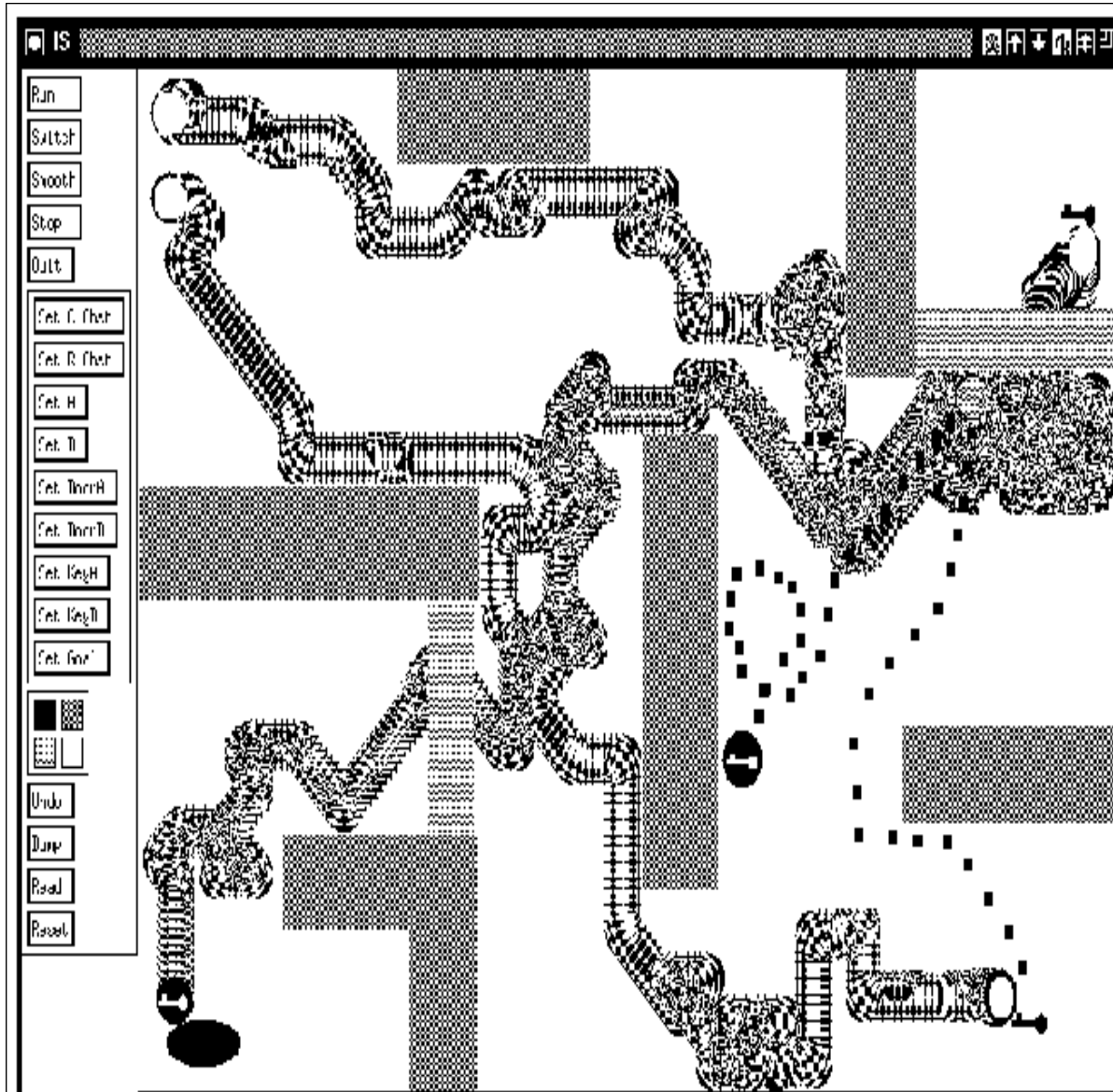












1997: Lifelong meta-learning with **self-modifying policies and success-story algorithm**: 2 agents, 2 doors, 2 keys. 1st southeast wins 5, the other 3. **Through recursive self-modifications only**: from 300,000 steps per trial down to 5,000.

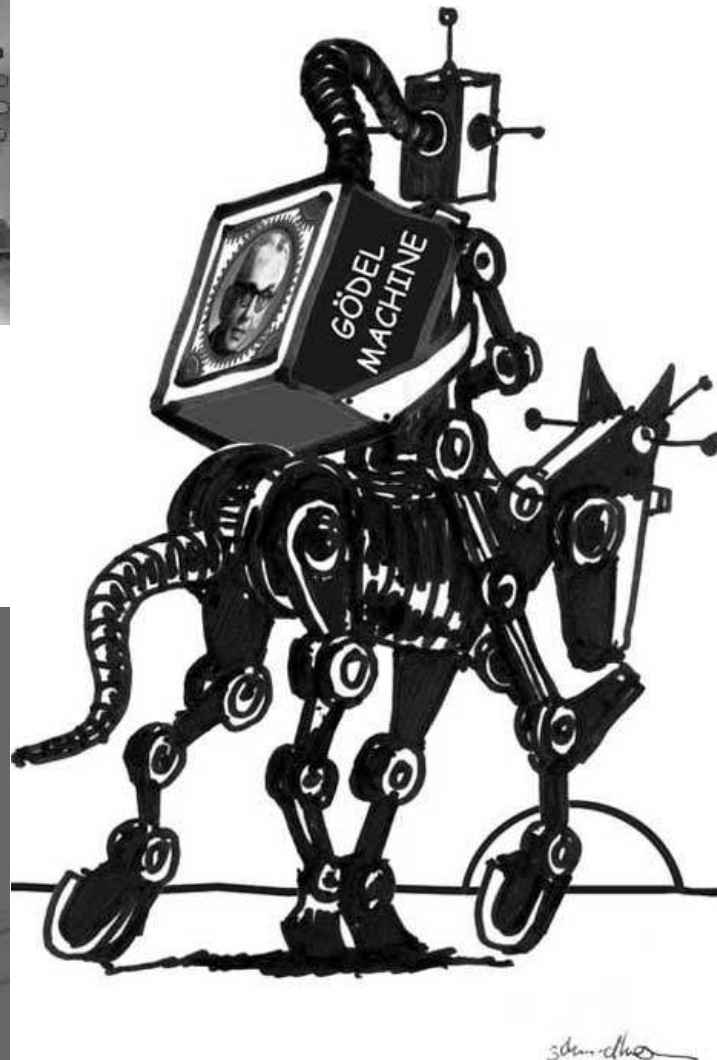
Kurt Gödel, father of theoretical computer science, exhibited the limits of math and computation (1931) by creating a formula that speaks about itself, claiming to be unprovable by a computational theorem prover: either formula is true but unprovable, or math is flawed in an algorithmic sense

Universal problem solver Gödel machine uses self reference trick in a new way





goedelmachine.com

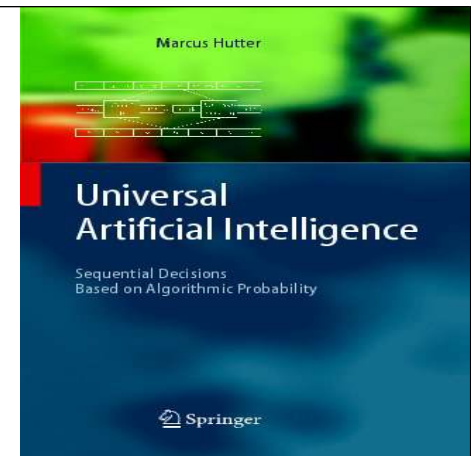


Gödel Machine (2003):
agent-controlling **program**
that speaks about itself,
ready to rewrite itself in
arbitrary fashion once it
has found a proof that the
rewrite is **useful**, given a
user-defined utility function

Theoretically optimal
self-improver!

Initialize Gödel Machine
by Marcus Hutter's
asymptotically fastest
method for all well-
defined problems

IDSIA
2002
on my
SNF
grant



Given $f:X \rightarrow Y$ and $x \in X$, search proofs to find
program q that provably computes $f(z)$ for all
 $z \in X$ within time bound $t_q(z)$; spend most time
on $f(x)$ -computing q with best current bound

As fast as fastest
 f -computer, save
for factor $1+\varepsilon$ and
 f -specific const.
independent of x !

$$n^3 + 10^{1000} = n^3 + O(1)$$

PowerPlay not only solves but also continually invents problems at the borderline between what's known and unknown - training an increasingly general problem solver by continually searching for the simplest still unsolvable problem

POWER PLAY

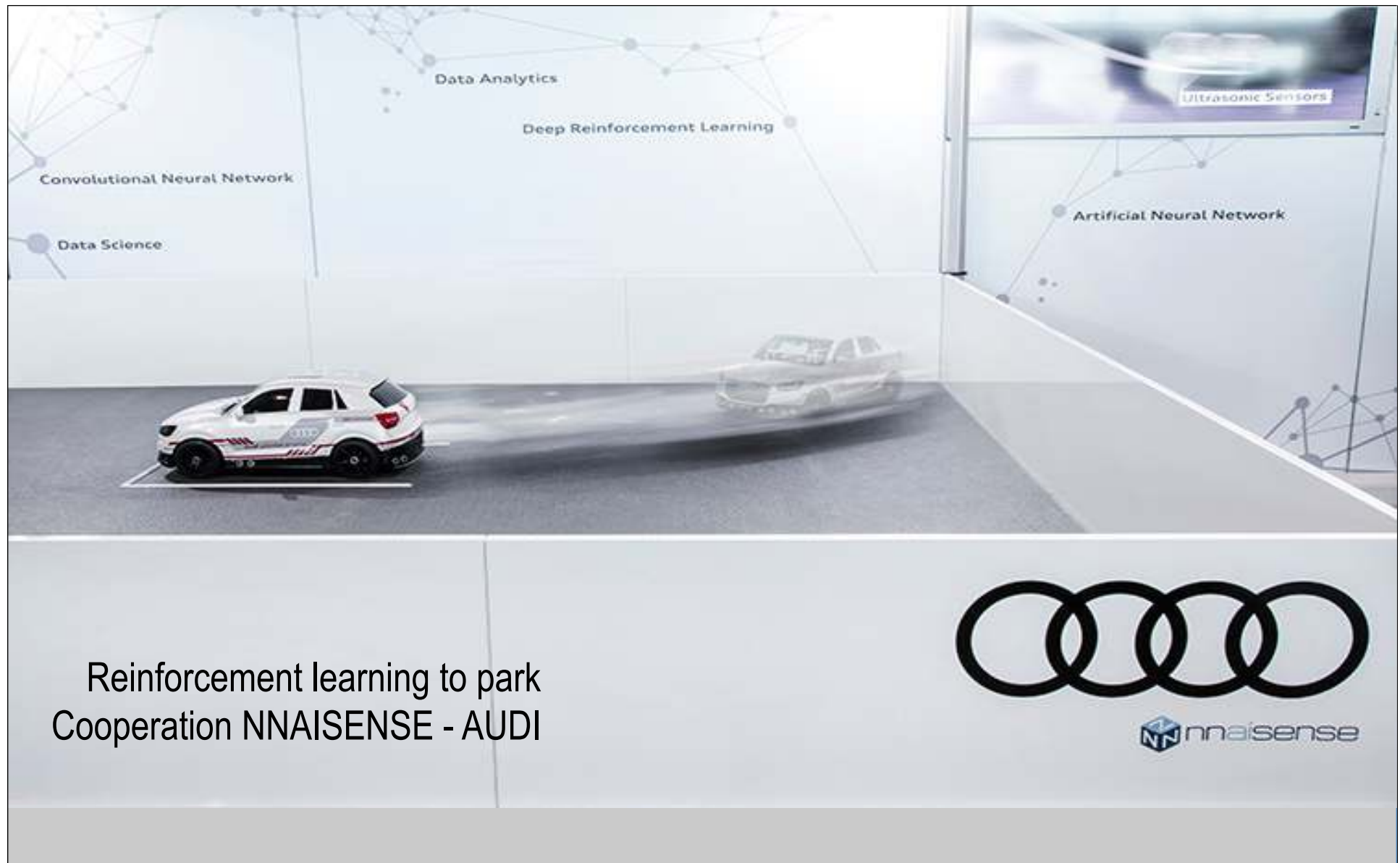




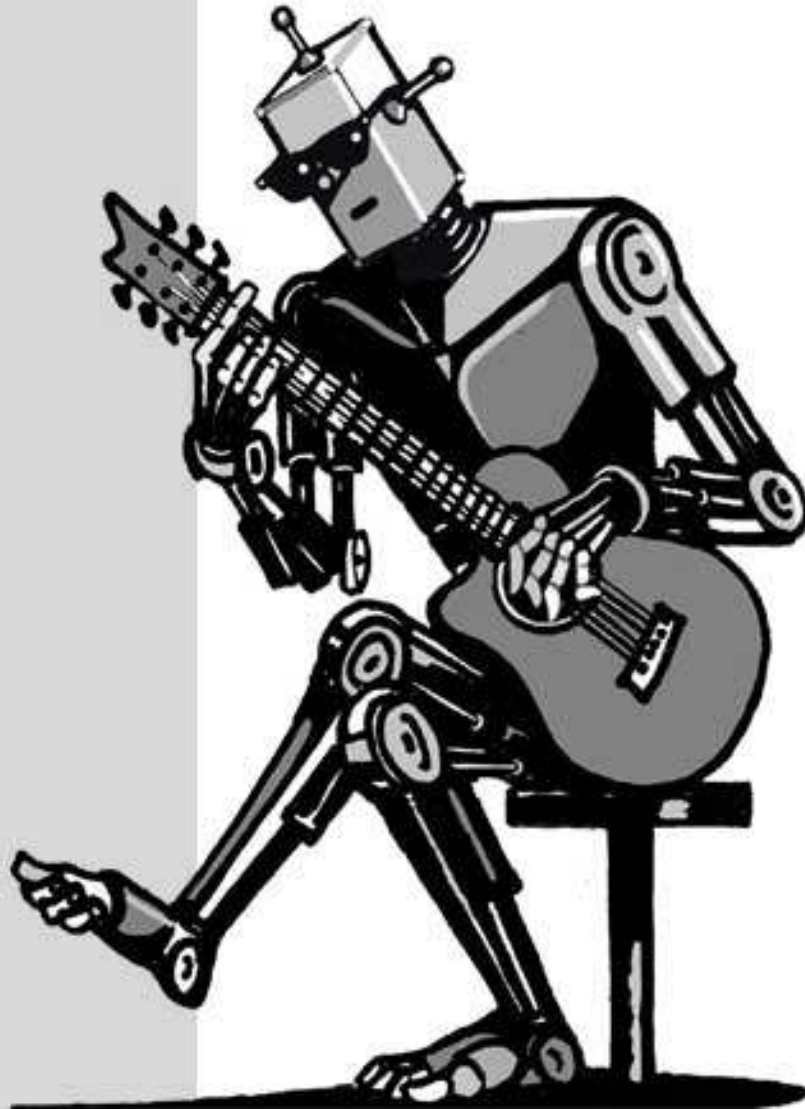
nnaisense

neural networks-based
artificial intelligence

THE DAWN OF AI



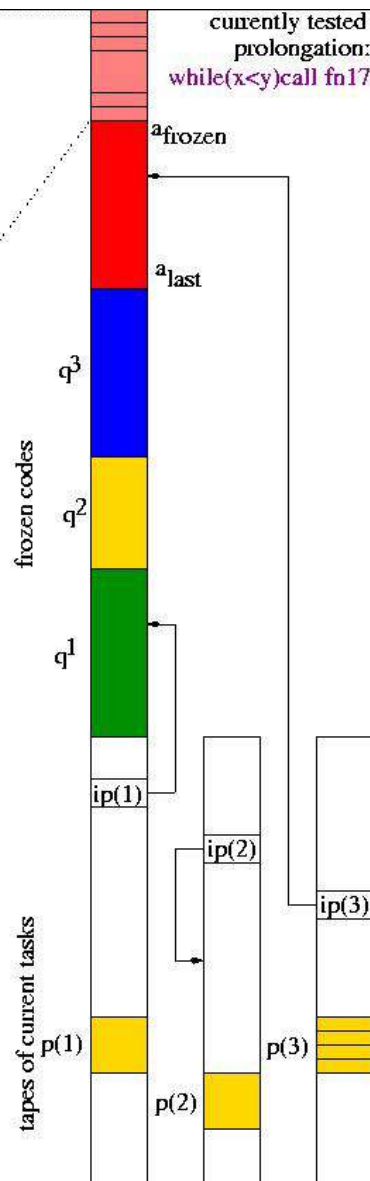
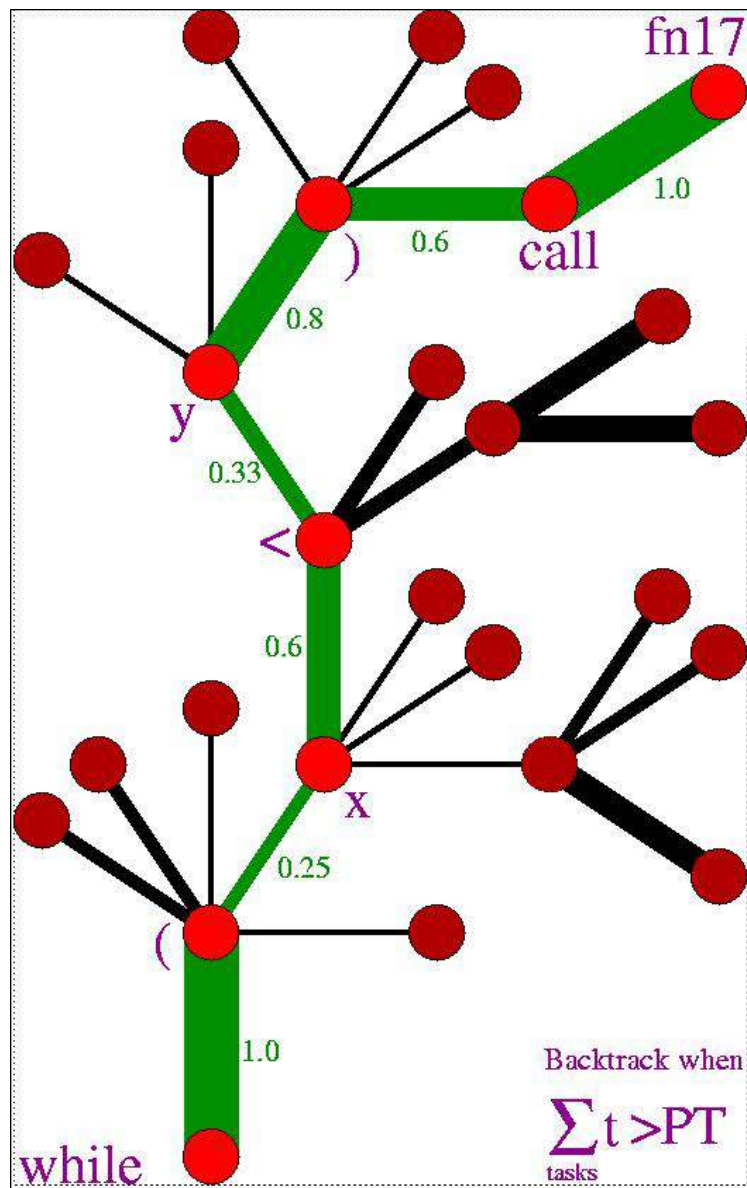
1. J. Schmidhuber. [Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook](#). Diploma thesis, TUM, 1987. (First concrete RSI.)
2. J. Schmidhuber. [A self-referential weight matrix](#). ICANN 1993
3. J. Schmidhuber. [On learning how to learn learning strategies](#). TR FKI-198-94, 1994.
4. J. Schmidhuber and J. Zhao and M. Wiering. [Simple principles of metalearning](#). TR IDSIA-69-96, 1996. (Based on 3.)
5. J. Schmidhuber, J. Zhao, N. Schraudolph. [Reinforcement learning with self-modifying policies](#). In *Learning to learn*, Kluwer, pages 293-309, 1997. (Based on 3.)
6. J. Schmidhuber, J. Zhao, and M. Wiering. [Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement](#). Machine Learning 28:105-130, 1997. (Based on 3.)
7. J. Schmidhuber. [Gödel machines: Fully Self-Referential Optimal Universal Self-Improvers](#). In *Artificial General Intelligence*, p. 119-226, 2006. (Based on TR of 2003.)
8. T. Schaul and J. Schmidhuber. [Metalearning](#). Scholarpedia, 5(6):4650, 2010.
9. More under <http://people.idsia.ch/~juergen/metalearner.html>



Learning how to Learn Learning Algorithms: Extra Slides

Jürgen Schmidhuber
The Swiss AI Lab IDSIA
Univ. Lugano & SUPSI
<http://www.idsia.ch/~juergen>

NNAISENSE

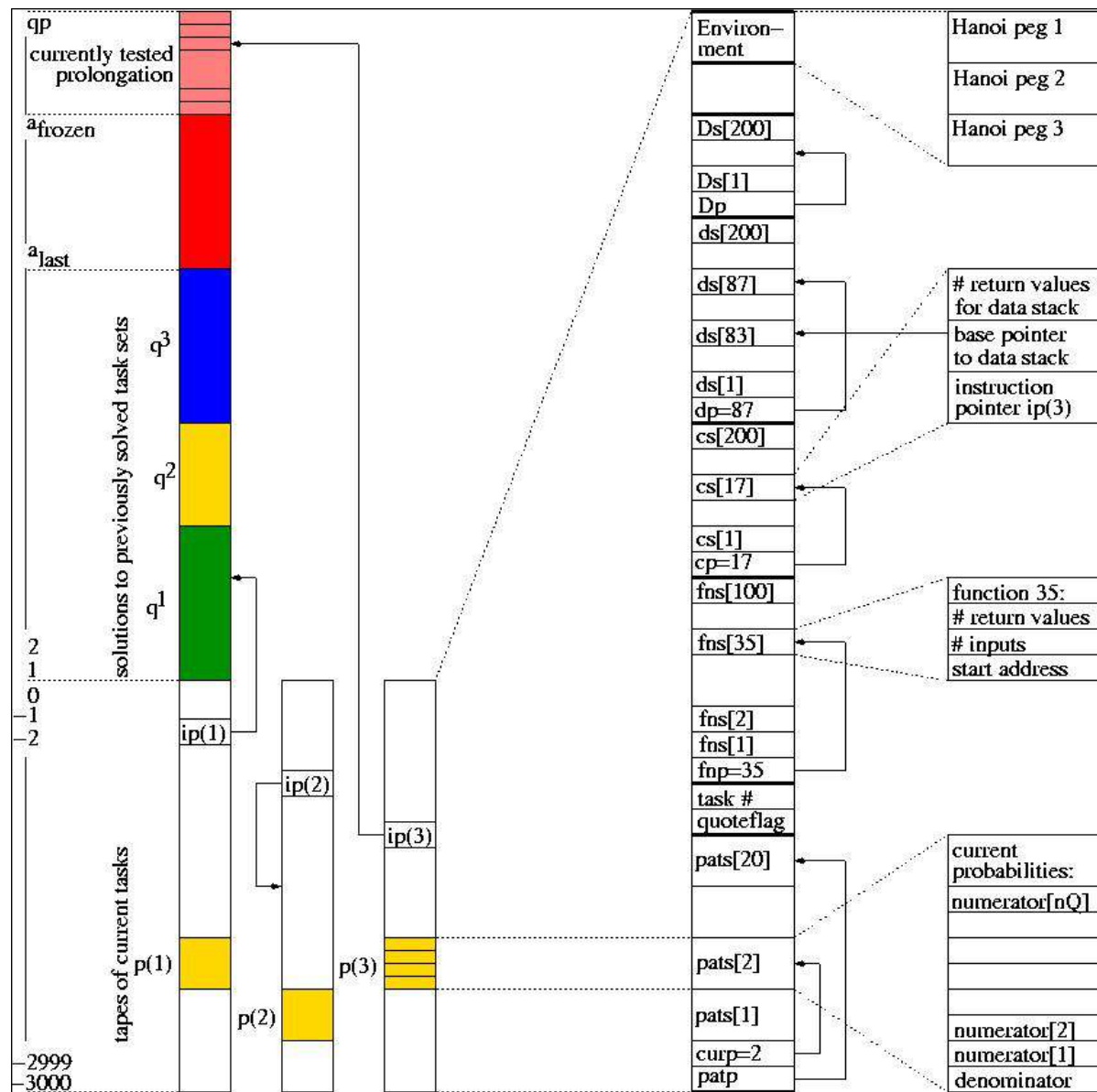


Super-deep program learner:
Optimal Ordered Problem Solver
OOPS (Schmidhuber, MLJ, 2004,
extending Levin's universal
search, 1973)

Time-optimal incremental search
and algorithmic transfer learning
in program space

Branches of search tree are
program prefixes

Node-oriented backtracking
restores partially solved task sets
& modified memory components
on error or when $\sum t > PT$



61 primitive instructions operating on stack-like and other internal data structures. For example:

push1(), not(x), inc(x), add(x,y), div(x,y), or(x,y), exch_stack(m,n), push_prog(n), movstring(a,b,n), delete(a,n), find(x), define function(m,n), callfun(fn), jumpif(val,address), quote(), unquote(), boost_probability(n,val)

Programs are integer sequences; data and code look the same; makes functional programming easy

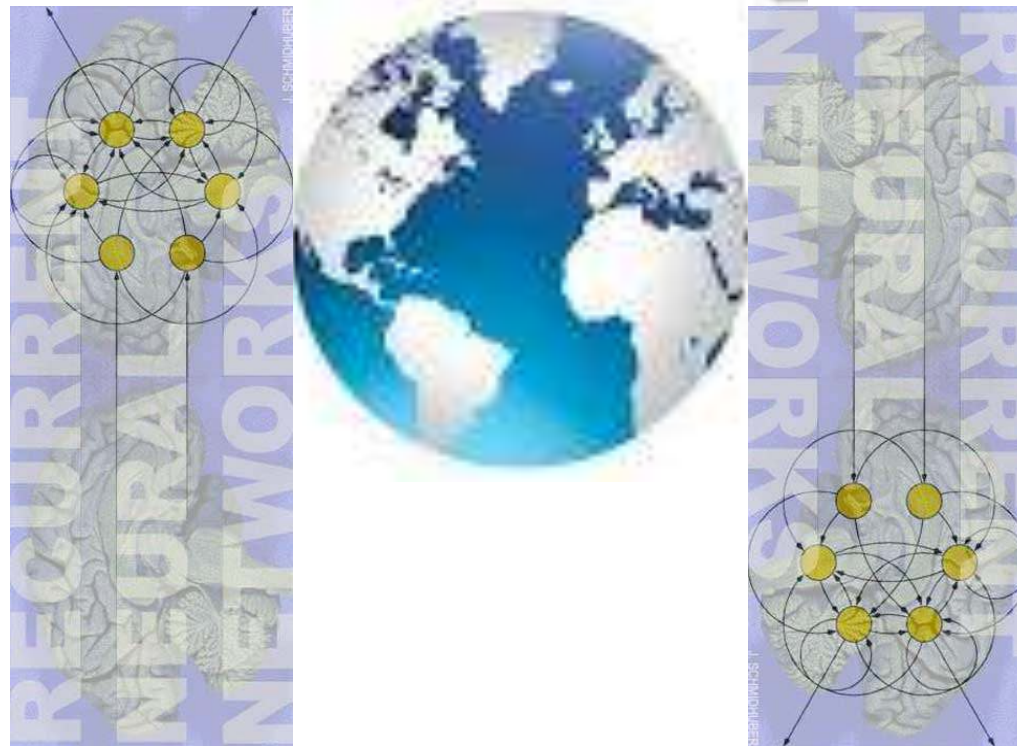
Towers of Hanoi: incremental solutions

- +1ms, $n=1$: *(movdisk)*
- 1 day, $n=1,2$: *(c4 c3 cpn c4 by2 c3 by2 exec)*
- 3 days, $n=1,2,3$: *(c3 dec boostq defnp c4 calltp c3 c5 calltp endnp)*
- 4 days: $n=4, n=5, \dots, n=30$: *by same double-recursive program*
- Profits from 30 earlier context-free language tasks ($1^n 2^n$): *transfer learning*
- 93,994,568,009 prefixes tested
- 345,450,362,522 instructions
- 678,634,413,962 time steps
- longest single run: 33 billion steps (5% of total time)! Much deeper than recent memory-based “deep learners” ...
- top stack size for restoring storage: < 20,000

What the found **Towers of Hanoi** solver does:

- (c3 dec boostq defnp c4 calltp c3 c5 calltp endnp)
- Prefix increases P of double-recursive procedure:
Hanoi(Source,Aux,Dest,n): IF n=0 exit; ELSE BEGIN
Hanoi(Source,Dest,Aux,n-1); move top disk from Aux to Dest;
Hanoi(Aux,Source,Dest,n-1); END
- Prefix boosts instructions of previously frozen program, which happens to be a previously learned solver of a context-free language ($1^n 2^n$). This rewrites search procedure itself: **Benefits of metalearning!**
- Prefix probability 0.003; suffix probability $3 \cdot 10^{-8}$; total probability $9 \cdot 10^{-11}$
- Suffix probability without prefix execution: $4 \cdot 10^{-14}$
- That is, Hanoi does profit from $1^n 2^n$ experience and incremental learning (OOPS excels at algorithmic transfer learning): speedup factor 1000

J.S.: IJCNN 1990, NIPS 1991: Reinforcement Learning with Recurrent Controller & Recurrent World Model

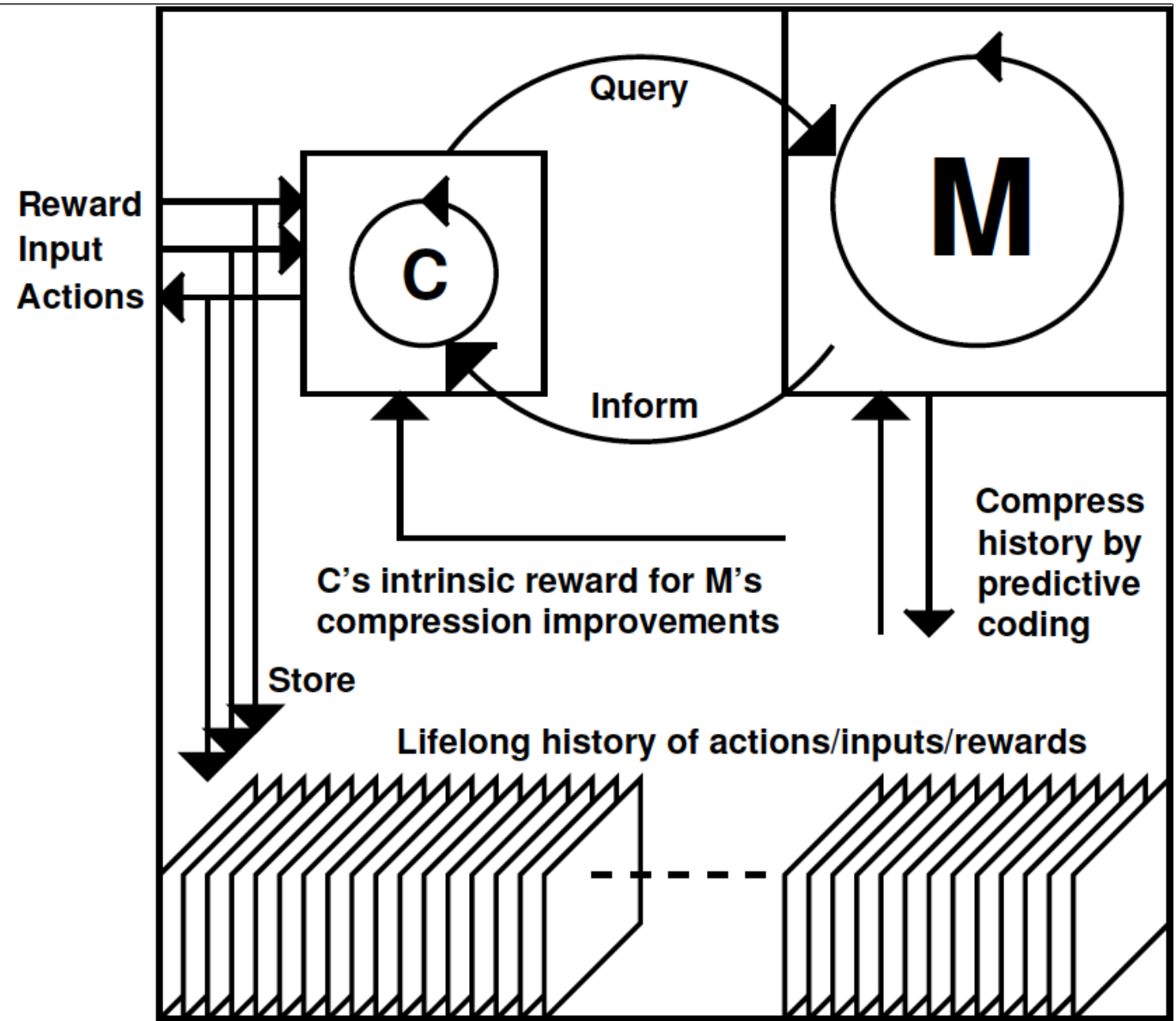


Learning
and
planning
with
recurrent
networks

RNNAlssance
2014-2015

On Learning to
Think: Algorithmic
Information
Theory for Novel
Combinations of
Reinforcement
Learning RNN-
based Controllers
(RNNAls) and
Recurrent Neural
World Models

<http://arxiv.org/abs/1511.09249>



[illegible]