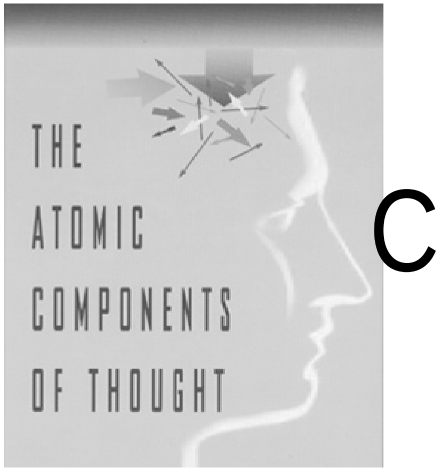


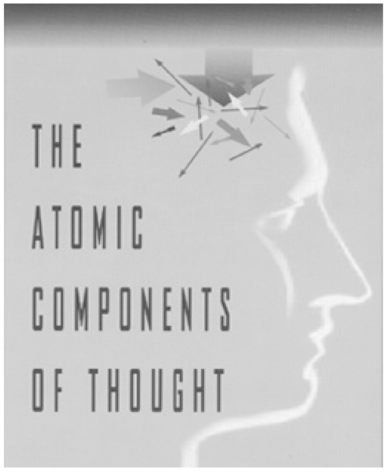
Compilation and Instruction

ACT-R Post Graduate
Summer School 2001

Coolfont Resort

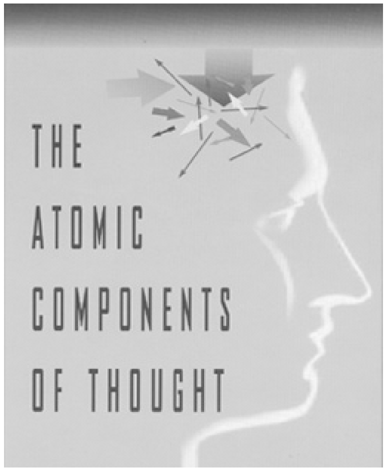
John R. Anderson
Psychology Department
Carnegie Mellon University
Pittsburgh, PA 15213
ja+@cmu.edu

ACT-R Home Page: http://act.psy.cmu.edu



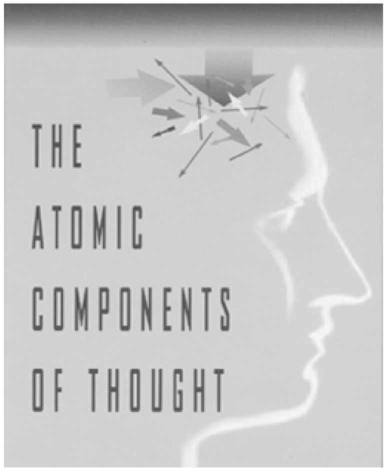
Notes on Compilation and Instruction

The missing elements to have a self-generating system.



Production Compilation: The Basic Idea

```
(p read-stimulus
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value =val
==>
   +retrieval>
      isa goal
      relation associate
      arg1 =val
      arg2 =ans
   =goal>
      relation associate
      arg1 =val
      step testing)

(p recall
   =goal>
      isa goal
      relation associate
      arg1 =val
      step testing
   =retrieval>
      isa goal
 relation associate
      arg1 =val
      arg2 =ans
==>
   +manual>
      isa press-key
      key =ans
   =goal>
      step waiting)

(p recall-vanilla
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value "vanilla
==>
   +manual>
      isa press-key
      key "7"
   =goal>
      relation associate
      arg1 "vanilla"
      step waiting)
```



Production Compilation: The Principles

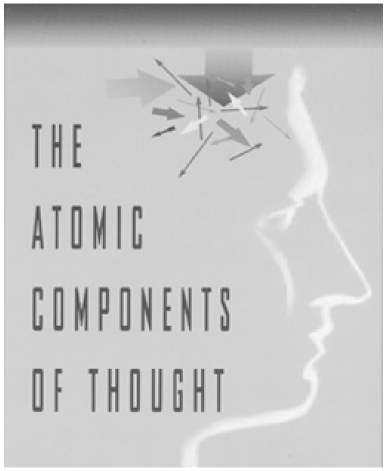
1. **Perceptual-Motor Buffers:** Avoid compositions that will result in jamming when one tries to build two operations on the same buffer into the same production.

2. **Retrieval Buffer:** Except for failure tests proceduralize out and build more specific productions.

3. **Goal Buffers:** Complex Rules describing merging.

4. **Safe Productions:** Production will not produce any result that the original productions did not produce.

5. **Parameter Setting:**
Successes = P***initial-experience***
Failures = (1-P) ***initial-experience***
Efforts = (Successes + Efforts)(C + *cost-penalty*)



Production Compilation: The Successes

1. **Taatgen:** Learning of inflection (English past and German plural). Shows that production compilation can come up with generalizations.

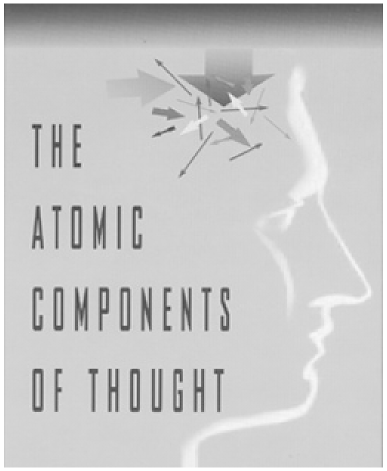
2. **Taatgen:** Learning of air-traffic control task – shows that production compilation can deal with complex perceptual motor skill.

3. **Anderson:** Learning of productions for performing paired associate task from instructions. Solves mystery of where the productions for doing an experiment come from.

4. **Anderson:** Learning to perform an anti-air warfare coordinator task from instructions. Shows the same as 2 & 3.

5. **Anderson:** Learning in the fan effect that produces the interaction between fan and practice. Justifies a major simplification in the parameterization of productions – no strength separate from utility.

Note all of these examples involve all forms of learning occurring in ACT-R simultaneous – acquiring new chunks, acquiring new productions, activation learning, and utility learning.



Proof of Concept
From Last Year’s Summer School:
Learning from Instruction

The Problems

-- modeling natural language comprehension

-- representing the product of comprehension

-- interpreting the representation

The Prolog Solution?

-- skip natural language

-- represent the instruction as a set of Prolog clauses
(not unique) that represent the knowledge in the instruction

-- encode in ACT-R a Prolog interpreter

-- each prolog clause corresponds to a goal and a unit task

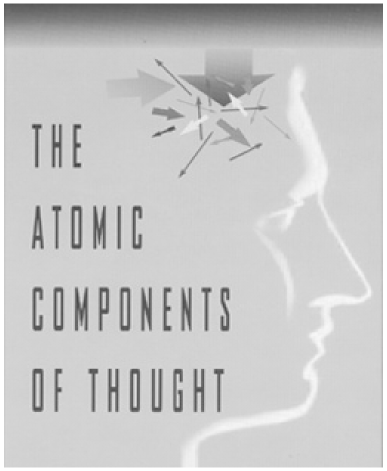
-- the thorny issue of backup

What does the Prolog Solution Represent?

--Not that Prolog is the right internal representation

--Rather that we have an outline for learning from
instruction if we have an adequately expressive
internal language.

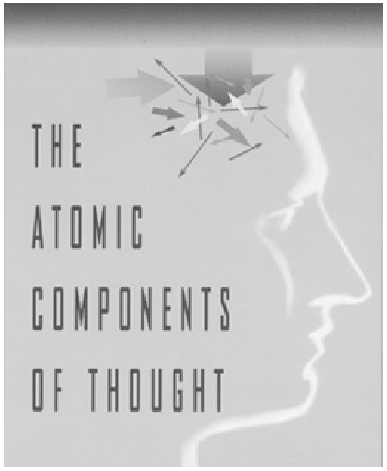
Now we are moving to a serious model of instruction representation and interpretation.



Paired Associate Example

	Data		Without Compilation		With Compilation	
Trial	Latency (sec.)	Accuracy	Latency (sec.)	Accuracy	Latency (sec.)	Accuracy
2	2.158	0.526	2.293	0.660	2.118	0.550
3	1.967	0.667	2.159	0.755	1.748	0.725
4	1.762	0.798	2.114	0.770	1.667	0.775
5	1.680	0.887	2.218	0.800	1.607	0.830
6	1.552	0.924	2.209	0.840	1.628	0.900
7	1.467	0.958	2.279	0.885	1.487	0.845
8	1.402	0.954	2.244	0.910	1.529	0.890

Without compilation latency is largely determined by the competition which stays relatively constant.



The “Instructions” for the Paired-Associate Task

1. To do the experiment you are to read the **stimulus**, associate the **stimulus** with the **response**, act on the **response**, and repeat.

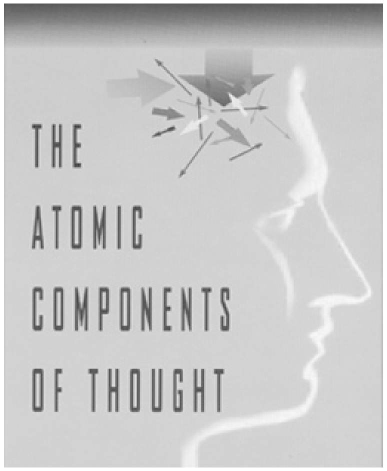
2. To associate a **response** with a **stimulus**, wait and read the **response**.

3. To act on an **item**, if you are still the stimulus stage, type the **item**, and read the **answer**.

4. Otherwise to act on an item just pass.

The “Prolog” clauses are:

```
do-experiment :- read(Stimulus),associate(Stimulus,Response),
                         act(Response), repeat.
associate(Probe,Answer):- read(Answer).
act(Item):- still-stimulus?, type (Item), read (Answer).
act(Item).
```



The Critical Features of Instructions and their Interpretation

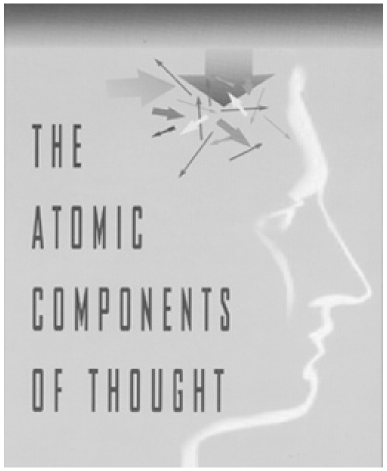
1. A rule for a goal is represented as an ordered sequence of clauses. Should it be not possible to satisfy one clause, the rule immediately fails. There is no backup.

2. The rules for a goal are tried in strict sequence so that default rules are tried only after special case rules.

3. Iteration is achieved with a special case repeat goal.

4. The terms capitalized above are variables. Rules have at most two variables.

5. Relations have at most two arguments.



The Actual Encoding for ACT-R

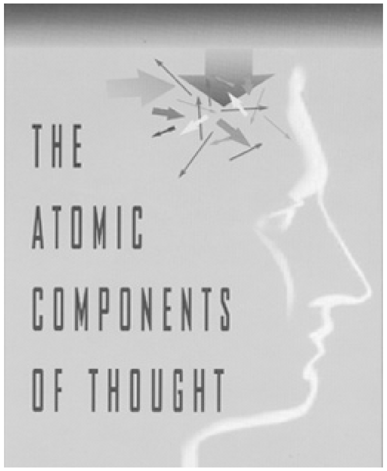
```
(setf instructions '(
    (do-experiment read (stimulus) associate (stimulus response)
                act (response) repeat)
    (associate (probe answer) read (answer) done)
    (act (item) still-stimulus? type (item) read (answer) done)
    (act (item) done)))
```

(parse instructions)

produces

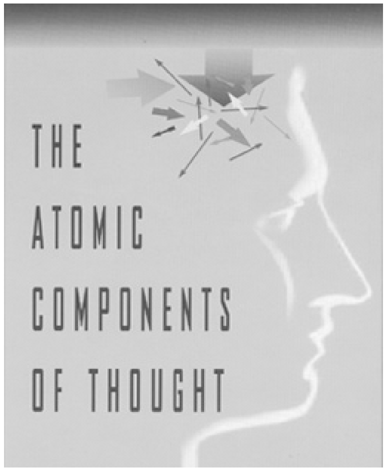
```
(RULE102 ISA HEAD RELATION DO-EXPERIMENT PRIOR START)
(P102 ISA CLAUSE RELATION READ PRIOR RULE102 ARG1 VAR1)
(P103 ISA CLAUSE RELATION ASSOCIATE PRIOR P102 ARG1 VAR1 ARG2 VAR2)
(P104 ISA CLAUSE RELATION ACT PRIOR P103 ARG1 VAR2)
(P105 ISA CLAUSE RELATION REPEAT PRIOR P104)
(RULE105 ISA HEAD RELATION ASSOCIATE PRIOR START ARG1 VAR1 ARG2 VAR2)
(P106 ISA CLAUSE RELATION READ PRIOR RULE105 ARG1 VAR2)
(P107 ISA CLAUSE RELATION done PRIOR P106)
(RULE106 ISA HEAD RELATION ACT PRIOR START ARG1 VAR1)
(P108 ISA CLAUSE RELATION STILL-STIMULUS? PRIOR RULE106)
(P109 ISA CLAUSE RELATION TYPE PRIOR P107 ARG1 VAR1)
(P110 ISA CLAUSE RELATION READ PRIOR P108 ARG1 VAR2)
(P111 ISA CLAUSE RELATION done PRIOR P110)
(RULE109 ISA HEAD RELATION ACT PRIOR RULE102 ARG1 VAR1)
(P111 ISA CLAUSE RELATION done PRIOR RULE109)
```

```
(chunk-type clause relation arg1 arg2 prior)
(chunk-type head relation arg1 arg2 prior)
(chunk-type task parent relation arg1 arg2 rule clause step var1 var2)
```



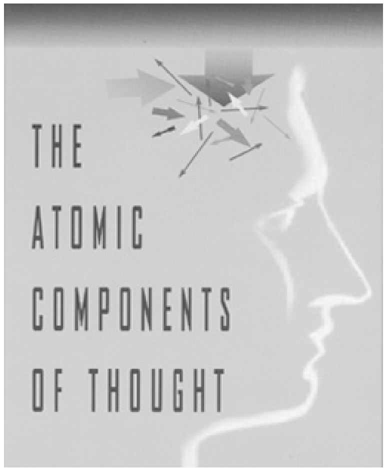
Initiation of a Rule

```
(p retrieve-rule
   =goal>
        isa task
        relation =relation
        step achieve
==>
    +retrieval>
       isa head
       relation =relation
       prior start
  =goal>
       step rule)
```



Instantiation of a 2-argument Head

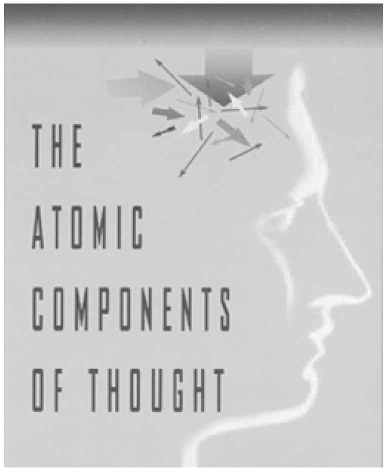
```
(p instantiate-rule-var1-var2
   =goal>
        isa task
        relation =relation
        arg1 =val1
        arg2 =val2
        step rule
    =retrieval>
       isa head
       arg1 var1
       arg2 var2
==>
     +retrieval>
         isa clause
         prior =retrieval
    =goal>
         step done
        relation nil
       arg1 nil
         arg2 nil
         var1 =val1
         var2 =val2
         rule =retrieval)
```



Backup

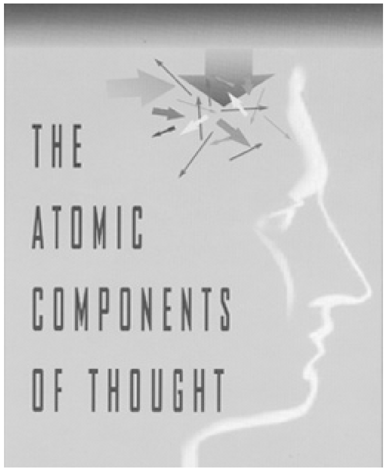
```
(p retry-higher
    =goal>
        isa task
        parent =parent
      - parent experiment
        step rule
    =retrieval>
        isa error
==>
   +retrieval>
         =parent
   =goal>
        step pop-failure)

(p pop-failure
    =goal>
        isa task
        step pop-failure
    =retrieval>
        isa task
        parent =grandparent
        rule =rule
==>
   +retrieval>
       =grandparent
   +goal>
        isa task
        step try-again
        rule =rule
        parent =grandparent)
```

```
(p retry-1-arg
    =goal>
        isa task
        step try-again
        rule =rule
    =retrieval>
        isa task
        relation =rel
        arg1 =arg1
        arg2 nil
==>
  +retrieval>
         isa head
         prior =rule
   =goal>
        relation =rel
        arg1 =arg1
        arg2 nil
        step rule)
```



Special Instructions for Achieving a Clause

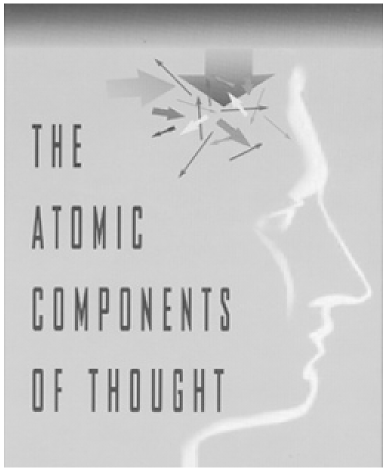
```
(p type-var1
     =goal>
    isa task
    step done
    var1 =val
     =retrieval>
         isa clause
         relation type
         arg1 var1
         arg2 nil
!eval! (equal (length =val) 1)
==>
   +manual>
      isa press-key
      key =val
    =goal>
       relation nil
       arg1 nil
       clause =retrieval
         step done
     +retrieval>
        isa clause
        prior =retrieval)
```



Retrieval to Instantiate a Clause

```
(p retrieve-*var1-var2
    =goal>
         isa task
         step done
         var1 =val1
         var2 nil
     =retrieval>
         isa clause
         relation =relation
         arg1  var1
         arg2 var2
==>
   =goal>
       relation =relation
       arg1 =val1
       arg2 var2
       clause =retrieval
       step retrieval-harvest
   +retrieval>
       isa task
       relation =relation
       arg1 =val1
     - arg2 var2
  -    step retrieval-harvest)
```

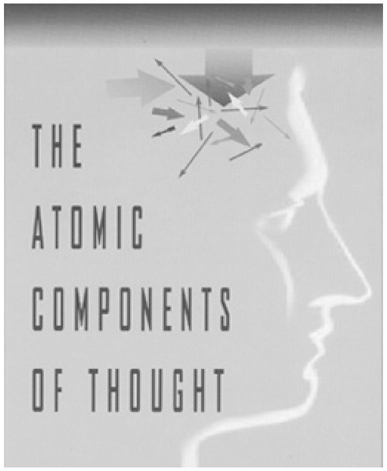
```
(p retrieve-*var2-var1
    =goal>
         isa task
         step done
         var1 nil
         var2 =val2
     =retrieval>
         isa clause
         relation =relation
         arg1 var2
         arg2  var1
==>
   =goal>
       relation =relation
       arg1 =val2
       arg2 var1
       clause =retrieval
       step retrieval-harvest
   +retrieval>
       isa task
       relation =relation
       arg1 =val2
     - arg2 var1
  -    step retrieval-harvest)
```



Harvesting Results

```
(p harvest-var2b
     =goal>
    isa TASK
    arg2 var2
    step retrieval-harvest
    clause =clause
    =retrieval>
         isa task
        arg2 =val
==>
    =goal>
        step done
        relation nil
        arg1 nil
        arg2 nil
        var2 =val
     +retrieval>
        isa clause
        prior =clause)
```

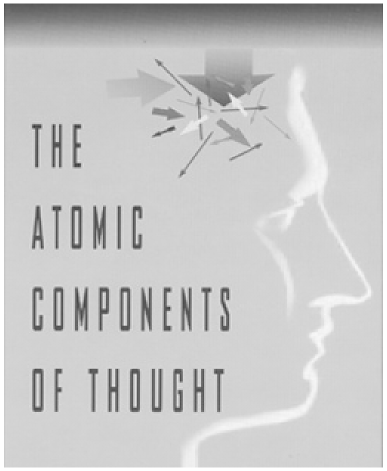
```
(p fail-harvest-var
     =goal>
          isa TASK
          relation =relation
          step retrieval-harvest
     =retrieval>
          isa error
==>
    +retrieval>
        isa head
        relation =relation
        prior start
     =goal>
        step subgoal-var)
```



Subgoaling

```
(p subgoal-find-second-arg
     =goal>
    isa TASK
    relation =relation
    arg1 =val
  - arg1 var1
  - arg1 var2
    step subgoal-var
    parent =parent
     =retrieval>
         isa head
         arg1 var1
         arg2 var2
==>
    +retrieval>
       isa clause
       prior =retrieval
   +goal>
       isa task
       var1 =val
       parent =goal
       step done
       rule =retrieval
   =goal>
       step subgoaled)

(p Go-Back-1
    =goal>
       isa TASK
       step Done
       parent =oldgoal
     - parent experiment
    =retrieval>
       isa clause
       relation done
 ==>
    +retrieval>
       =oldgoal
    =goal>
       step go-back)
```

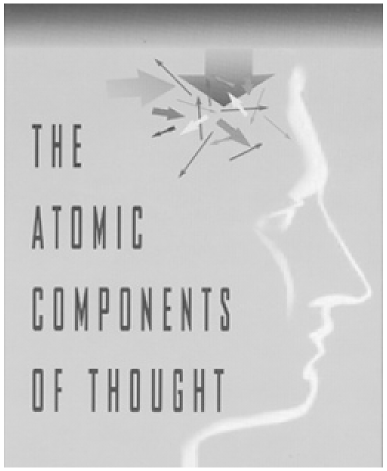
```
(p go-back-arg1b
   =goal>
       isa task
       step go-back
       var1 =arg1
     =retrieval>
         isa TASK
         relation =rel
         arg1 var2
         arg2 =arg2
         step Subgoaled
==>
  =goal>
     relation =rel
     arg1 =arg1
     arg2 =arg2
    +retrieval>
       =goal
    +goal>
       =retrieval)

(p harvest-subgoal-var2a
     =goal>
    isa TASK
    arg2 var2
    step Subgoaled
    clause =clause
   =retrieval>
    isa task
    var2 =val
==>
  +retrieval>
    isa clause
    prior =clause
  =goal>
    arg2 =val
    var2 =val
    step done)
```



Subgoaling for Side Effects

```
(p subgoal-2-given-var2
    =goal>
        isa task
        step done
        var2 =val2
     =retrieval>
        isa clause
        relation =relation
        arg1 =val1
      -  arg1 var1
      - arg1 var2
        arg2  var2
==>
   =goal>
       relation =relation
       arg1 =val1
       arg2 =val2
       step subgoaled
       clause =retrieval
   +goal>
       isa task
       relation =relation
       arg1 =val1
       arg2 =val2
       parent =goal
       step achieve)
```

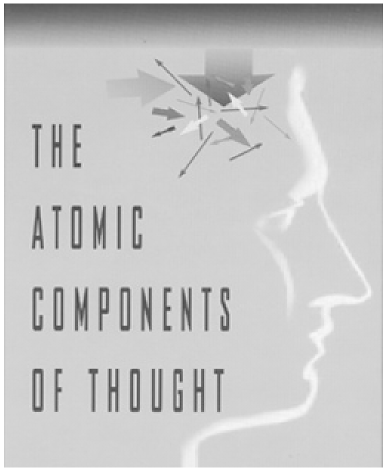
```
(p go-back-side-effect
   =goal>
      isa task
      step go-back
    =retrieval>
   isa TASK
    - arg1 var1
   - arg1 var2
    - arg2 var1
    - arg2 var2
   step Subgoaled
==>
    +retrieval>
       =goal
    +goal>
       =retrieval)

(p forward-subgoal-default
    =goal>
        isa task
        step subgoaled
        clause =clause
        - arg1 var1
        - arg1 var2
        - arg2 var1
        - arg2 var2
==>
    =goal>
       step done
       relation nil
       arg1 nil
       arg2 nil
     +retrieval>
       isa clause
       prior =clause)
```



Repeating

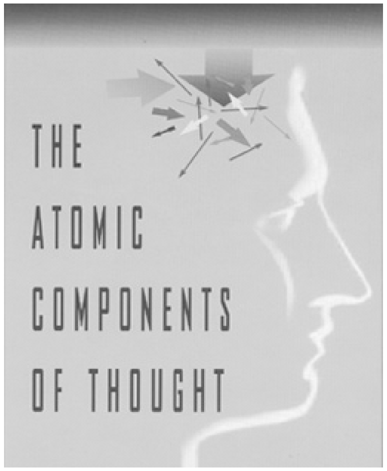
```
(p repeat-1-arg
     =goal>
          isa task
          parent =parent
          step repeat
          rule =rule
     =retrieval>
          isa task
          relation =rel
          arg1 =arg1
          arg2 nil
==>
   +retrieval>
           isa head
           relation =rel
           prior start
    +goal>
          isa task
          relation =rel
          arg1 =arg1
          arg2 nil
          step rule
          parent =parent)
```

```
(p repeat
     =goal>
           isa task
           step done
           parent =parent
    =retrieval>
        isa clause
    relation repeat
==>
   =goal>
        step repeat
    +retrieval> =parent)
```



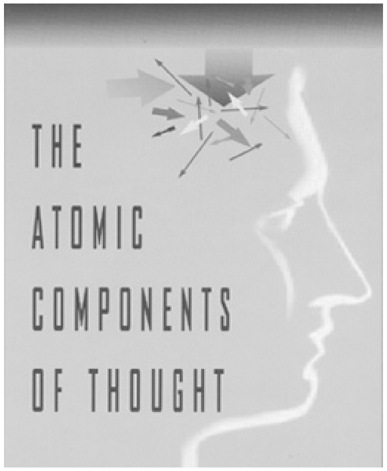
A Trial without Production Compilation

```
Time 70.000: * Running stopped because time limit reached
 Time 70.083: P498149 Retrieved
 Time 70.083: Ready-To-Read Selected
 Time 70.133: Ready-To-Read Fired
 Time 70.133: Read-Attend Selected
 Time 70.183: Read-Attend Fired
 Time 70.183: Module :VISION running command MOVE-ATTENTION
 Time 70.233: Module :VISION running command FOCUS-ON
 Time 70.233: Read-Bind-Var1 Selected
 Time 70.283: Read-Bind-Var1 Fired
 Time 70.452: P498150 Retrieved
 Time 70.452: Retrieve-*Var1-Var2 Selected
 Time 70.502: Retrieve-*Var1-Var2 Fired
 Time 71.405: Goal498174 Retrieved
 Time 71.405: Harvest-Var2b Selected
 Time 71.455: Harvest-Var2b Fired
 Time 71.607: P498151 Retrieved
 Time 71.607: Subgoal-1-Var2 Selected
 Time 71.657: Subgoal-1-Var2 Fired
 Time 71.657: Retrieve-Rule Selected
 Time 71.707: Retrieve-Rule Fired
 Time 71.836: Rule498154 Retrieved
 Time 71.836: Instantiate-Rule-Var1 Selected
 Time 71.886: Instantiate-Rule-Var1 Fired
 Time 71.941: P498155 Retrieved
 Time 71.941: Still-Stimulus Selected
 Time 71.991: Still-Stimulus Fired
 Time 72.140: P498156 Retrieved
 Time 72.140: Type-Var1 Selected
 Time 72.190: Type-Var1 Fired
 Time 72.190: Module :MOTOR running command PRESS-KEY
 Time 72.290: Module :MOTOR running command PREPARATION-COMPLETE
 Time 72.335: P498157 Retrieved
 Time 72.335: Ready-To-Read Selected
 Time 72.385: Ready-To-Read Fired
 Time 72.440: Device running command OUTPUT-KEY
```



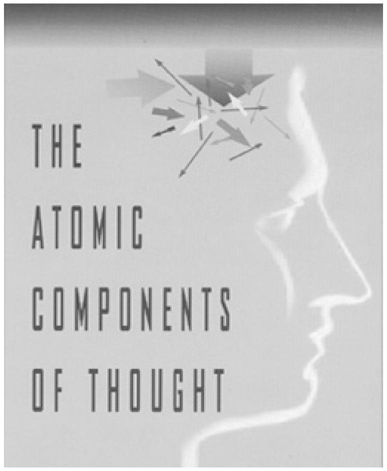
A Trial after the Point of Maximal Learning

```
Time 70.000: * Running stopped because time limit reached
Time 70.000: Read-Attend Selected
Time 70.050: Read-Attend Fired
Time 70.050: Module :VISION running command MOVE-ATTENTION
Time 70.100: Module :VISION running command FOCUS-ON
Time 70.100: Production24197 Selected
Time 70.150: Production24197 Fired
Time 70.150: Module :MOTOR running command PRESS-KEY
Time 70.250: Module :MOTOR running command PREPARATION-COMPLETE
Time 70.400: Device running command OUTPUT-KEY
```



The Learned Production

```
(p Production24197
   =goal>
      isa TASK
      arg1 Var1
      relation Read
      step Reading
      clause P24143
      var2 nil
   =visual>
      isa TEXT
      value "zinc"
   !eval! (stimulus =goal)
==>
   -visual-location>
   =goal>
      relation Act
      arg1 "9"
      arg2 nil
      step Subgoaled
      clause P24145
      var2 "9"
      var1 "zinc"
   +manual>
      isa PRESS-KEY
      key "9"
   +goal>
      isa TASK
      relation Read
      arg1 Var2
      arg2 nil
      clause P24151
      step Ready-To-Read
      rule Rule24148
      var1 "9"
      parent =goal)
```



Instructions for the Athena Task

```
 start :- change-radius(128), id.
change-radius(X) :- select(display), select(radius),
                         select(X), select(execute).
id :- find-closest(X), id-sequence(X), repeat.
find-closest(X) :- seek(anzio), attend-closest(X),
                       mouse, hook(X).
idsequence(X) :- altitude, test, speed-test,
                     classify(arinc).
idsequence(X) :- ews(X,T), classify(T).
classify(T) :- match(T,arinc),idit(friend,non-military).
idit(X,Y) :- select(track), select(update), select(class),
             select(primary),select(X),select(air),select(save).
ews(X,T) :-select(ews),select(query),identity(T).
altitutde-test :- seek(upper-left),search-down(alt,W),
                  read-next(W,Z),<(Z,40000),<(20000,Z).
speed-test :- seek(upper-left),search-down(speed,W),
                  read-next(W,Z),<(Z,500),<(350,Z).
select(T) :- find-menu(T,L),key(L).
find-menu(T,L) :- seek(lower-left),search-right(T,L).
key(L) :- find-count(L,C),f-key(C).
f-key(C) :- append-F(C,K),hit(K).
search-right(T,L) :- match-right(T), current(L).
match-right(T) :- read-right(I),match(T,I).
match-right(T) :- repeat.
search-down(T,L):- match-down(T), current(L).
match-down(T) :- read-down(I),match(T,I).
match-down(T) :- repeat.
find-count(L,N):- seek(lower-left),attend-button,
                       init-count(1), count-to(L,N).
count-to(L,N) :- at(L),count(N).
count-to(L,N) :- next(button), increment-count.
```