Backward Chainer

Technical Presentation

Nil Geisweiller

OpenCog Foundation

Mini CogAthon 2017

Forward vs Backward Chaining

- Forward: build inferences from sources to targets
- Backward: build inferences from targets to sources

Backward Chaining as Forward Control

Backward Chaining can be seen as searching forward chaining control strategies that may prove a target.

Backward Chainer evolves an and-or-tree representing forward chaining control strategies

- Root: target
- Vertices: premises of parents / conclusions of children
- Edges: rules connecting premises to conclusion
- Leaves: axioms in the knowledge base

This and-or-tree is called the Back-Inference Tree (BIT).

Why and-or-tree?

- and-branches: all premises of a rule must be fulfilled
- or-branches: multiple rules can prove the same target

In practice the BIT is stored as a collection of and-BITs obtained by following certain paths along or-branches.

Examples of and-BITs

1.
```
[10187631544213947954][78] [10892703157338374946][78]
----------------bc-deduction-formula-----------------
              [10437095963339794776][1]
```

2.
```
              [17701765486026989748][1] [15171751518105380644][78]
              -------conditional-full-instantiation-formula-------
[10187631544213947954][78] [10892703157338374946][78]
----------------bc-deduction-formula-----------------
              [10437095963339794776][1]
```

3.
```
                           [11343694068128076800][1] [17258454523025295511][1]
                           ======fuzzy-conjunction-introduction-formula=======
             [13836867826147951221][1] [14074315650070651326][62]
             -------conditional-full-instantiation-formula-------
[17436618117399753180][1] [12750438441371402854][62]
-------conditional-full-instantiation-formula-------
             [12943974154720736373][1]
```

4.
```
[14240257604799279553][78] [14264241660366004022][78] [14592143235347104002][78] [14953255393560133990][78]
==================================fuzzy-conjunction-introduction-formula==================================
[17701765486026989748][1]              [17766956938503895142][78]
--------------conditional-full-instantiation-formula---------------
                   [10437095963339794776][1] [9901656951636544264][78]
                   ---------------bc-deduction-formula----------------
                                [10437095963339794776][1]
```

5.
```
             [17843132057269021466][1] [15900832770610379591][1] [10812302800942215856][1] [18030308772378534194][1]
             ------conditional-full-instantiation-formula------- ----------------bc-deduction-formula----------------
[10739589484482059731][1] [17216414254437066572][1]                          [15765921954083210084][1] [13980066437165088146][1]
=============================================fuzzy-conjunction-introduction-formula=============================================
[17701765486026989748][1]                              [10836755343021939996][1]
-------------------conditional-full-instantiation-formula--------------------
                         [98250191651577789 20][1]
```

High Level Algorithm

1. Expand BIT
   - 1.1 Select and-BIT, leaf and rule
   - 1.2 Expand and-BIT from leaf with rule
2. Fulfill BIT
   - 2.1 Select and-BIT
   - 2.2 Execute its Forward Chaining Control Strategy
   - 2.3 Collect results if any
3. Reduce BIT
   - 3.1 Select and-BITs
   - 3.2 Remove them from the BIT
4. Goto 1 (unless termination)

High Level Algorithm

1. Expand BIT
   - 1.1 Select and-BIT, leaf and rule
   - 1.2 Expand and-BIT from leaf with rule
2. Fulfill BIT
   - 2.1 Select and-BIT
   - 2.2 Execute its Forward Chaining Control Strategy
   - 2.3 Collect results if any
3. Reduce BIT
   - 3.1 Select and-BITs
   - 3.2 Remove them from the BIT
4. Goto 1 (unless termination)

Rule Format

```
BindLink
  <variables>
  AndLink                    \
    <pattern-1>              |
    ...                      |
    <pattern-n>              | Find premises in the atomspace
    <precondition-1>         |
    ...                      |
    <precondition-m>         /
  ExecutionOutputLink        \
    <formula>                |
    ListLink                 |
      <conclusion>           | Produce conclusions with them
      <premise-1>            |
      ...                    |
      <premise-k>            /
```

Rule Example: Modus Ponens: A, A->B |- B

```
(BindLink
  (VariableList                                          \
    (TypedVariableLink                                   |
      (VariableNode "$A")                                |
      (TypeChoice                                        |
        (TypeNode "LambdaLink")                          |
        (TypeNode "PredicateNode")                       |
      )                                                  |
    )                                                    | Variables
    (TypedVariableLink                                   |
      (VariableNode "$B")                                |
      (TypeChoice                                        |
        (TypeNode "LambdaLink")                          |
        (TypeNode "PredicateNode")                       |
      )                                                  |
    )                                                    |
  )                                                      /
  (AndLink                                               \
    (ImplicationLink                                     |
      (VariableNode "$A")                                | Pattern
      (VariableNode "$B")                                |
    )                                                    /
    (EvaluationLink                                      \
      (GroundedPredicateNode "scm: true-enough")         |
      (ImplicationLink                                   |
        (VariableNode "$A")                              |
        (VariableNode "$B")                              |
      )                                                  |
    )                                                    | Preconditions
    (EvaluationLink                                      |
      (GroundedPredicateNode "scm: true-enough")         |
      (VariableNode "$A")                                |
    )                                                    |
  )                                                      /
  (ExecutionOutputLink                                   \
    (GroundedSchemaNode "scm: crisp-modus-ponens-formula") |
    (ListLink                                            |
      (VariableNode "$B")                                |
      (VariableNode "$A")                                |
      (ImplicationLink                                   | Production
        (VariableNode "$A")                              |
        (VariableNode "$B")                              |
      )                                                  |
    )                                                    |
  )                                                      /
)
```

Rule Example: Deduction: A->B, B->C |- A->C

```
(BindLink
  (VariableList                                          \
    (TypedVariableLink                                   |
      (VariableNode "$A")                                |
      (TypeNode "ConceptNode")                           |
    )                                                    |
    (TypedVariableLink                                   |
      (VariableNode "$B")                                |
      (TypeNode "ConceptNode")                           | Variables
    )                                                    |
    (TypedVariableLink                                   |
      (VariableNode "$C")                                |
      (TypeNode "ConceptNode")                           |
    )                                                    |
  )                                                      /
  (AndLink                                               \
    (InheritanceLink                                     |
      (VariableNode "$B")                                |
      (VariableNode "$C")                                |
    )                                                    | Patterns
    (InheritanceLink                                     |
      (VariableNode "$A")                                |
      (VariableNode "$B")                                |
    )                                                    /
    (EvaluationLink                                      \
      (GroundedPredicateNode "scm: true-enough")         |
      (InheritanceLink                                   |
        (VariableNode "$A")                              |
        (VariableNode "$B")                              |
      )                                                  |
    )                                                    |
    (EvaluationLink                                      | Preconditions
      (GroundedPredicateNode "scm: true-enough")         |
      (InheritanceLink                                   |
        (VariableNode "$B")                              |
        (VariableNode "$C")                              |
      )                                                  |
    )                                                    |
  )                                                      /
  (ExecutionOutputLink                                   \
    (GroundedSchemaNode "scm: bc-deduction-formula")     |
    (ListLink                                            |
      (InheritanceLink                                   |
        (VariableNode "$A")                              |
        (VariableNode "$C")                              |
      )                                                  |
      (InheritanceLink                                   | Production
        (VariableNode "$A")                              |
        (VariableNode "$B")                              |
      )                                                  |
      (InheritanceLink                                   |
        (VariableNode "$B")                              |
        (VariableNode "$C")                              |
      )                                                  |
    )                                                    /
  )
)
```

Expand and-BIT Example: Who's the Criminal?

Query:

```
(cog-bc
  rb                              <- Rule base
  (InheritanceLink                \
    (VariableNode "$who")         | target
    (ConceptNode "criminal"))     /
  (TypedVariableLink              \
    (VariableNode "$who")         | variable declaration
    (TypeNode "ConceptNode"))     /
)
```

Expand and-BIT Example: Who's the Criminal?

Query:

```
(cog-bc
  rb                              <- Rule base
  (InheritanceLink                \
    (VariableNode "$who")         | target
    (ConceptNode "criminal"))     /
  (TypedVariableLink              \
    (VariableNode "$who")         | variable declaration
    (TypeNode "ConceptNode"))     /
)
```

Initial and-BIT:

```
   (BindLink
     (TypedVariableLink
       (VariableNode "$who") ; [7437519479921537920][1]
       (TypeNode "ConceptNode") ; [3788625745177846543][1]
     ) ; [11334884684560273995][1]
     (InheritanceLink
       (VariableNode "$who") ; [7437519479921537920][1]
       (ConceptNode "criminal") ; [4346836709706211120][1]
     ) ; [10437095963339794776][1]
     (InheritanceLink
       (VariableNode "$who") ; [7437519479921537920][1]
       (ConceptNode "criminal") ; [4346836709706211120][1]
     ) ; [10437095963339794776][1]
   ) ; [17025922688105268679][78]
```

[10437095963339794776][1]

Expand and-BIT Example: Who's the Criminal?

Query:

```
(cog-bc
  rb                              <- Rule base
  (InheritanceLink                \
    (VariableNode "$who")         | target
    (ConceptNode "criminal"))     /
  (TypedVariableLink              \
    (VariableNode "$who")         | variable declaration
    (TypeNode "ConceptNode"))     /
)
```

Initial and-BIT:

```
(BindLink
  (TypedVariableLink
    (VariableNode "$who") ; [7437519479921537920][1]
    (TypeNode "ConceptNode") ; [3788625745177846543][1]
  ) ; [11334884684560273995][1]
  (InheritanceLink
    (VariableNode "$who") ; [7437519479921537920][1]
    (ConceptNode "criminal") ; [4346836709706211120][1]
  ) ; [10437095963339794776][1]
  (InheritanceLink
    (VariableNode "$who") ; [7437519479921537920][1]
    (ConceptNode "criminal") ; [4346836709706211120][1]
  ) ; [10437095963339794776][1]
) ; [17025922688105268679][78]
```

[10437095963339794776][1]

Expand and-BIT Example: Who's the Criminal?

Unify the deduction rule with the target.

```
(BindLink
  (VariableList
    (TypedVariableLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11334884684560273995][1]
    (TypedVariableLink
      (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11790491878558854165][78]
  ) ; [17749752181343036404][78]
  (AndLink
    (InheritanceLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
    ) ; [10187631544213947954][78]
    (InheritanceLink
      (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      (ConceptNode "criminal") ; [4346836709706211120][1]
    ) ; [10892703157338374946][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      ) ; [10187631544213947954][78]
    ) ; [16948020608718095165][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
        (ConceptNode "criminal") ; [4346836709706211120][1]
      ) ; [10892703157338374946][78]
    ) ; [17653092221842522157][78]
  ) ; [14412449543575778263][78]
  (ExecutionOutputLink
    (GroundedSchemaNode "scm: bc-deduction-formula") ; [548150114326654866][1]
    (ListLink
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (ConceptNode "criminal") ; [4346836709706211120][1]
      ) ; [10437095963339794776][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      ) ; [10187631544213947954][78]
      (InheritanceLink
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
        (ConceptNode "criminal") ; [4346836709706211120][1]
      ) ; [10892703157338374946][78]
    ) ; [12675979575027112735][78]
  ) ; [11205444531943140963][78]
) ; [14867172891917739103][18446744073709551615]
```

Expand and-BIT Example: Who's the Criminal?

Expand pattern term: substitute terms involving the target by the rule pattern

```
(InheritanceLink
  (VariableNode "$who")
  (ConceptNode "criminal")
)
```

```
(AndLink
  (InheritanceLink
    (VariableNode "$who")
    (VariableNode "$B-6266d6f2")
  )
  (InheritanceLink
    (VariableNode "$B-6266d6f2")
    (ConceptNode "criminal")
  )
  (EvaluationLink
    (GroundedPredicateNode "scm: true-enough")
    (InheritanceLink
      (VariableNode "$who")
      (VariableNode "$B-6266d6f2")
    )
  )
  (EvaluationLink
    (GroundedPredicateNode "scm: true-enough")
    (InheritanceLink
      (VariableNode "$B-6266d6f2")
      (ConceptNode "criminal")
    )
  )
)
```

Expand and-BIT Example: Who's the Criminal?

Expand rewrite term: substitute the target by the rule rewrite

```
(InheritanceLink
  (VariableNode "$who")
  (ConceptNode "criminal")
)
```

```
(ExecutionOutputLink
  (GroundedSchemaNode "scm: bc-deduction-formula")
  (ListLink
    (InheritanceLink
      (VariableNode "$who")
      (ConceptNode "criminal")
    )
    (InheritanceLink
      (VariableNode "$who")
      (VariableNode "$B-6266d6f2")
    )
    (InheritanceLink
      (VariableNode "$B-6266d6f2")
      (ConceptNode "criminal")
    )
  )
)
```

Expand and-BIT Example: Who's the Criminal?

```
(BindLink
  (VariableList
    (TypedVariableLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11334884684560273995][1]
    (TypedVariableLink
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11790491878558854165][78]
  ) ; [17749752181343036404][78]
  (AndLink
    (InheritanceLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
    ) ; [10187631544213947954][78]
    (InheritanceLink
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      (ConceptNode "criminal") ; [43468367097062111 20][1]
    ) ; [10892703157338374946][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      ) ; [10187631544213947954][78]
    ) ; [16948020608718095165][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
        (ConceptNode "criminal") ; [43468367097062111 20][1]
      ) ; [10892703157338374946][78]
    ) ; [17653092221842522157][78]
  ) ; [14412449543575778263][78]
  (ExecutionOutputLink
    (GroundedSchemaNode "scm: bc-deduction-formula") ; [5481501143266548866][1]
    (ListLink
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (ConceptNode "criminal") ; [43468367097062111 20][1]
      ) ; [10437095963339794776][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      ) ; [10187631544213947954][78]
      (InheritanceLink
        (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
        (ConceptNode "criminal") ; [43468367097062111 20][1]
      ) ; [10892703157338374946][78]
    ) ; [12675979575027112735][78]
  ) ; [11205444531943140963][78]
) ; [14867172891917739103][78]
```

```
[10187631544213947954][78] [10892703157338374946][78]
----------------bc-deduction-formula-----------------
              [10437095963339794776][1]
```

Expand and-BIT Example: Who's the Criminal?

```
(BindLink
  (VariableList
    (TypedVariableLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11334884684560273995][1]
    (TypedVariableLink
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [11790491878558854165][78]
    (TypedVariableLink
      (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
      (TypeNode "ConceptNode") ; [3788625745177846543][1]
    ) ; [176025736149570025 62][78]
  ) ; [13048585240571552182][78]
  (AndLink
    (InheritanceLink
      (VariableNode "$who") ; [7437519479921537920][1]
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
    ) ; [10187631544213947954][78]
    (InheritanceLink
      (VariableNode "$B-6266d6f2") ; [40973722905803642 98][78]
      (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
    ) ; [15850292629115077001][78]
    (InheritanceLink
      (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
      (ConceptNode "criminal") ; [4346836709706211120][1]
    ) ; [16704784893736523343][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
        (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
      ) ; [15850292629115077001][78]
    ) ; [13387309656764448404][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
        (ConceptNode "criminal") ; [4346836709706211120][1]
      ) ; [16704784893736523343][78]
    ) ; [14241801921385894746][78]
    (EvaluationLink
      (GroundedPredicateNode "scm: true-enough") ; [4909270440978081812][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      ) ; [10187631544213947954][78]
    ) ; [16948020608718095165][78]
  ) ; [9965343418475005290][78]
  (ExecutionOutputLink
    (GroundedSchemaNode "scm: bc-deduction-formula") ; [5481501143266548866][1]
    (ListLink
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (ConceptNode "criminal") ; [4346836709706211120][1]
      ) ; [10437095963339794776][1]
      (InheritanceLink
        (VariableNode "$who") ; [7437519479921537920][1]
        (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
      ) ; [10187631544213947954][78]
      (ExecutionOutputLink
        (GroundedSchemaNode "scm: bc-deduction-formula") ; [5481501143266548866][1]
        (ListLink
          (InheritanceLink
            (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
            (ConceptNode "criminal") ; [4346836709706211120][1]
          ) ; [10892703157338374946][78]
          (InheritanceLink
            (VariableNode "$B-6266d6f2") ; [4097372290580364298][78]
            (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
          ) ; [15850292629115077001][78]
          (InheritanceLink
            (VariableNode "$B-10cb7b10") ; [81054144628137367][78]
            (ConceptNode "criminal") ; [4346836709706211120][1]
          ) ; [16704784893736523343][78]
        ) ; [97592086135081392 13][78]
      ) ; [17512045607278943249][78]
    ) ; [10071949988112905230][78]
  ) ; [17824786981883709266][78]
) ; [10973664019817387519][78]
```

```
                [15850292629115077001][78] [16704784893736523343][78]
                ----------------bc-deduction-formula-----------------
[10187631544213947954][78] [10892703157338374946][78]
----------------bc-deduction-formula-----------------
                [10437095963339794776][1]
```

Random Selections

And-BITs, leaves and rules are selected according to probabilistic distributions.

- And-BITs are selected according to a simplicity bias.
- Leaves are selected according to their confidences.
- Rules are selected according to their weights.

Ultimately, these distributions will be modulated by meta-learning with the semantics that the probability of such selections may lead to a successful inference.