# Non-Axiomatic Reasoning System (NARS) solving the Facebook AI Research bAbI tasks

Patrick Hammer

June 21, 2015

**Abstract**

In "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks" [1, 2] 20 examples for AI-complete question answering were proposed, which an general AI system, according to them, has to be able to deal with. This paper is applying Pei Wang's Non-Axiomatic Reasoning System [3, 4, 5] to all of the proposed bAbI tasks.

## Contents

# 1 NAL syntax examples

We will express all knowledge to reason about in the so called Non-Axiomatic Logic (NAL), the cognitive logic which NARS uses for reasoning. How NAL works can be seen in detail in [5]. Here only the for the bAbI tasks needed aspects are described by a handful of examples:

A is a special case of B:

$$A \rightarrow B$$

A is an individual, not a category on its own:

$$\{A\}$$

A is a property:

$$[A]$$

$a_1, ..., a_n$ are related through relation:

$$(*, a_1, ..., a_n) \rightarrow relation$$

which can also be written as:

$$a_i \rightarrow (/, relation, a_1, ..., \_, ..., a_n)$$

Something unspecified which is a special case of B:

$$\#1 \rightarrow B$$

If something is a bird it is also flying:

$$(\$1 \rightarrow bird) \implies (\$1 \rightarrow [flying])$$

Such implications can also have time attached, like "if A happens then B happens at the same time":

```
A =|> B
```

and "B happens after A happens":

```
A =/> B
```

The time difference between A and B can also be measured, for example by "+10", written as:

```
(&/,A,+10) =/> B
```

Additionally each NAL-sentence will here either end with "." which represents a judgement, or end with "?", representing a question. Furthermore, each sentence with "." punctuation has an truth-value $< f, c >$ attached, where $f$ represents the frequency, which is defined as $\frac{w+}{w}$ where $w+$ is the positive evidence and $w$ the total evidence, and $c$ represents the confidence, which is defined as $\frac{w}{w+1}$. If unspecified the sentence has truth-value $< 1.0, 0.9 >$.

That something is not true, can in NARS either be represented as

```
J. <0.0,c>
```

or with negation as

```
(--,J). <1.0,c>
```

where J is a judgement.

Additionally the truth of all sentences can have an tense attached, : | : for "now", : / : for "in the future", and with backslash for "in the past", if none is specified it means "always". To note here is that "now" is not the same as 5 seconds later, reasoning time in NARS is not independent from real time.

# 2 Factoid QA

## 2.1 Basic Factoid QA with Single Supporting Fact

These are defined by [2] as basic question answering where a single supporting fact leads to the answer.

### 2.1.1 Example

John is in the playground.

```
((*,{john},{playground}) --> isin).
```

Bob is in the office.

```
((*,bob,office) --> isin).
```

Where is john?

```
((*,{john},{?where}) --> isin)?
```

A:playground

```
Answer ((*,{john},{playground}) --> isin). <1,0.9>
```

### 2.1.2 Answering Process

In this example all the system has to do is identifying

```
((*,{john},{playground}) --> isin).
```

as the answer to the question

```
((*,{john},{?where}) --> isin)?
```

by variable unfication, "?where:=playground".

## 2.2 Factoid QA with Two Supporting Facts

### 2.2.1 Example

Background knowledge: If something is picked, it means that the object which is picked is where the person is.

```
((&&,((*,$Person,$Object) --> pick),((*,$Person,$Place) --> isin)) =|>
((*,$Object,$Place) --> isin)).
```

John is in the playground.

```
((*,{john},{playground}) --> isin). :|:
```

Bob is in the office.

```
((*,{bob},{office}) --> isin). :|:
```

John picked up the football.

```
((*,{john},{football}) --> pick). :|:
```

Bob went to the kitchen.

```
((*,{bob},{kitchen}) --> isin). :|:
```

Where is the football?

```
((*,{football},{?where}) --> isin)?
```

A:playground

```
Answer ((*,{football},{playground}) --> isin). <1;0.28>
```

//Where was Bob before the kitchen?

```
((&/,((*,{bob},{?Where}) --> isin),?1) =/> ((*,{bob},{kitchen}) --> isin))?
```

A:office

```
Answer ((&/,((*,{bob},{office}) --> isin),+3) =/>
((*,{bob},{kitchen}) --> isin)). <1,0.31>
```

### 2.2.2 Answering Process

This one needs a form of temporal reasoning/perception, so that the system can see that

```
((*,{bob},{office}) --> isin). :|:
```

was indeed happening before

```
((*,{bob},{kitchen}) --> isin). :|:
```

happened. In this example it was through temporal induction supported by one example, indicating that "usually after bob is in the office, he is in the kitchen" Then like in the first example all what is left is recognizing

```
((&/,((*,{bob},{office}) --> isin),+3) =/> ((*,{bob},{kitchen}) --> isin)).
```

as the answer to the question.

# 3 Relations

## 3.1 Two Argument Relations: Subject vs. Object

### 3.1.1 Example

The office is north of the bedroom.

```
((*,{office},{bedroom}) --> northof).
```

The bedroom is north of the bathroom.

```
((*,{bedroom},{bathroom}) --> northof).
```

What is north of the bedroom?

```
((*,{?What},{bedroom}) --> northof)?
```

What is the bedroom north of?

```
((*,{bedroom},{?What}) --> northof)?
```

A:office

```
Answer ((*,{office},{bedroom}) --> northof). <1,0.9>
```

A:bathroom

```
Answer ((*,{bedroom},{bathroom}) --> northof). <1,0.9>
```

## 3.2 Three Argument Relations

### 3.2.1 Example

Mary gave the cake to Fred.

```
((*,{mary},{cake},{fred}) --> gave).
```

Fred gave the cake to Bill.

```
((*,{fred},{cake},{bill}) --> gave).
```

Jeff was given the milk by Bill.

```
((*,{bill},{milk},{jeff}) --> gave).
```

Who gave the cake to Fred?

```
((*,{?Who},{cake},{fred}) --> gave)?
```

Who did Fred give the cake to?

```
((*,{fred},{cake},{?Who}) --> gave)?
```

What did Jeff receive?

```
((*,{?1},{?WhatReceived},{jeff}) --> gave)?
```

Who gave the milk?

```
((*,{?Who},{milk},{?1}) --> gave)?
```

A:Mary

```
Answer ((*,{mary},{cake},{fred}) --> gave). <1,0.9>
```

A:Bill

```
Answer ((*,{fred},{cake},{bill}) --> gave). <1,0.9>
```

A:milk

```
Answer ((*,{bill},{milk},{jeff}) --> gave). <1,0.9>
```

A:Bill

```
Answer ((*,{bill},{milk},{jeff}) --> gave). <1,0.9>
```

### 3.3 Answering Process

In NARS arbitrary relations can be represented, this example again only demands variable unification / simple pattern matching.

## 4 Counting

These examples are about counting elements in a set.

### 4.1 Example

Background knowledge: What the count relation of elements in a set means

```
(({$1} --> $rel) ==> ((*,1,$rel,(*,$1)) --> count)).
(({$1,$2} --> $rel) ==>
(&&,(--,((*,1,$rel,(*,$1)) --> count)),((*,2,$rel,(*,$1,$2)) --> count))).
(({$1,$2,$3} --> $rel) ==> (&&,(--,((*,1,$rel,(*,$1)) --> count)),
(--,((*,2,$rel,(*,$1,$2)) --> count)),((*,3,$rel,(*,$1,$2,$3)) --> count))).
```

Input: Daniel picked up the football.

```
({football} --> (/,hold,{daniel},_)). :|:
```

Daniel dropped the football.

```
({football} --> (/,hold,{daniel},_)). :|: <0,0.9>
```

Daniel got the milk.

```
({milk} --> (/,hold,{daniel},_)). :|:
```

Daniel took the apple.

```
({apple} --> (/,hold,{daniel},_)). :|:
```

How many objects is Daniel holding?

```
((*,?HowMany,(/,hold,{daniel},_),?M) --> count)?
```

A:two, Only at least one, but under AIKR this is fine.

```
Answer ((*,1,(/,hold,{daniel},_),(*,football)) --> count). <1,0.45>
```

### 4.2 Answering Process

Here the question answering is more complicated. What does counting mean if it is not even entirely sure whether the elements to count fullfill the given relation or not? (insufficient knowledge) What does counting mean if the relation to check the data for demands, maybe also because of the amount of data, too much time and space? Like pointed out by Pei Wang in [3, 4] an intelligent system has to be able to deal with insufficient knowledge and resources (AIKR). This means, an AGI has to answer this questions only as good as the current knowledge and resources allow. This is also what happens above and in all other examples this paper demonstrates. Once answers are found, simple answers (low syntactic complexity) of high truth expectation (how true the judgement is) and originality (how much was considered) are preferred.

## 5 Lists/Sets

### 5.1 Example

These type of examples are about forming sets.

Daniel picks up the football.

```
({football} --> (/,hold,daniel,_)). :|:
```

Daniel drops the newspaper.

```
({newspaper} --> (/,hold,daniel,_)). :|: <0,0.9>
```

Daniel picks up the milk.

```
({milk} --> (/,hold,daniel,_)). :|:
```

What is Daniel holding?

```
({?What} --> (/,hold,daniel,_))?
({?What,?What2} --> (/,hold,daniel,_))?
({?What,?What2,?What3} --> (/,hold,{daniel},_))?
```

A:milk

```
Answer ({milk} --> (/,hold,daniel,_)). <1,0.47>
```

A:football,milk

```
Answer ({football,milk} --> (/,hold,daniel,_)). <1,0.3>
```

### 5.2 Answering Process

For such examples it is needed for NARS to be able to combine sets to bigger sets.

```
({a1,...,an} --> M).
({b1,...,bn} --> M).
|-
({a1,...,an,b1,...,bn} --> M).
```

The rest once again is pattern matching / variable unification.

## 6 Simple Negation

### 6.1 Example

Simple Negation Sandra travelled to the office.

```
((*,{sandra},{office}) --> at). :|:
```

Fred is no longer in the office.

```
((*,{fred},{office}) --> at). :|: <0,0.9>
```

Is Fred in the office?

```
((*,{fred},{office}) --> at)?
```

A:no, Fred was not in the office

```
Answer ((*,{fred},{office}) --> at). :\: <0,0.9>
```

Is Sandra in the office?

```
((*,{sandra},{office}) --> at)?
```

A:yes, Sandra was in the office

```
Answer ((*,{sandra},{office}) --> at). :\: <1,0.9>
```

### 6.2 Answering Process

Here the questions were yes/no question, directly answerable by recognizing the corresponding judgements as answers to the question, where negation was expressed like seen in the NAL syntax examples section.

## 7 Indefinite knowledges

Reasoning examples about the unknown.

### 7.1 Example

background knowledge: Johny can't be at the classroom or playground and at the same time in the office

```
(({john} --> (/,at,_,{classroom})) ==> (--,({john} --> (/,at,_,{office})))).
(({john} --> (/,at,_,{playground})) ==> (--,(john --> (/,at,_,{office})))).
```

John is either in the classroom or the playground.

```
({john} --> (/,at,_,{classroom})).
({john} --> (/,at,_,{playground})).
(({john} --> (/,at,_,{classroom})) ==> (--,({john} --> (/,at,_,{playground})))).
(({john} --> (/,at,_,{playground})) ==> (--,({john} --> (/,at,_,{classroom})))).
```

Sandra is in the garden.

```
({sandra} --> (/,at,_,{garden})).
```

Is John in the classroom?

```
({john} --> (/,at,_,{classroom}))?
```

Is John in the office?

```
({john} --> (/,at,_,{office}))?
```

A:maybe

```
Answer ({john} --> (/,at,_,{classroom})). <0.68,0.93>
```

A:no

```
Answer ({john} --> (/,at,_,{office})). <0,0.9>
```

### 7.2 Answering Process

Since the truth values of NAL distinguish between that something is not true (low frequency), and that something is not well known yet (low confidence), the above is possible. The first answer was that the system knows much about that it doesn't know about whether John is in the class room, and the second answer that the system knows much about that John is not in the office.

## 8 Coreference

### 8.1 Basic Coreference

#### 8.1.1 Example

Daniel was in the kitchen.

```
((*,{daniel},{kitchen}) --> at). :\:
```

Then he went to the studio.

```
(--,((*,{daniel},{kitchen}) --> at)). :|:
((*,{daniel},{studio}) --> at). :|:
```

Sandra was in the office.

```
((*,{sandra},{kitchen}) --> at). :\:
```

Where is Daniel? A:studio

```
((*,{daniel},?where) --> at)?
```

A:studio

```
Answer ((*,{daniel},{studio}) --> at). <1,0.47>
```

## 8.2 Compound Coreference

### 8.2.1 Example

Daniel and Sandra journeyed to the office.

```
({sandra} --> (/,at,_,{office})). :|:
({daniel} --> (/,at,_,{office})). :|:
```

Then they went to the garden.

```
({sandra} --> (/,at,_,{office})). :|: <0,0.9>
({daniel} --> (/,at,_,{office})). :|: <0,0.9>
({daniel} --> (/,at,_,{garden})). :|:
({sandra} --> (/,at,_,{garden})). :|:
```

Sandra and John travelled to the kitchen.

```
({sandra,john} --> (/,at,_,{kitchen})). :|:
```

After that they moved to the hallway.

```
({sandra,john} --> (/,at,_,{hallway})). :|:
```

Where is Daniel?

```
({daniel} --> (/,at,_,?Where))?
```

A:garden

```
Answer ({daniel} --> (/,at,_,{garden})). <1,0.47>
```

## 8.3 Answering Process

This examples implicitely assume, that it is known by the AI system, that something can't be at two places at the same time. This knowledge has to be given to NARS or acquired by observations by the system itself, or an implicit handling like above is also possible, whenever the position event comes also a corresponding "the object is not at the last position anymore" event is input to the system, this representation is the easiest for the system to work with.

# 9 Time manipulation

Examples which demand Temporal Reasoning in order to be understood.

## 9.1 Example

In the afternoon Julie went to the park.

```
((*,{julie},{park}) --> go). :|:
```

Yesterday Julie was at school.

```
((*,{julie},{school}) --> go). :\:
```

Julie went to the cinema this evening.

```
((*,{julie},{cinema}) --> go). :/:
```

Where did Julie go after the park?

```
((&/,((*,{julie},{park}) --> go),?1) =/> ((*,{julie},{?where}) --> go))?
```

A:cinema

```
Answer ((&/,((*,{julie},{park}) --> go),+3) =/> ((*,{julie},{cinema}) --> go)). <1,0.31>
```

## 9.2 Answering Process

This example demands temporal reasoning and variable unification, namely to form the temporal implication statement with temporal induction, and then recognize the answer as answer to the question. "Evening", "afternoon" etc. could also have been represented as a own event here, but this wasn't necessary in this case.

# 10 Basic Inference

## 10.1 Basic Deduction

Deriving known from known.

### 10.1.1 Example

Sheep are afraid of wolves.

```
((*,sheep,wolves) --> afraidof).
```

Cats are afraid of dogs.

```
((*,cats,dogs) --> afraidof).
```

Mice are afraid of cats.

```
((*,mice,cats) --> afraidof).
```

Gertrude is a sheep.

```
(gertrude --> sheep).
```

What is Gertrude afraid of? A:wolves

```
((*,gertrude,?what) --> afraidof)?
```

A:wolves

```
Answer ((*,gertrude,wolves) --> afraidof). <1,0.73>
```

## 10.2 Basic Induction

Generalizing using the induction principle and using the resulting hypothesis on a new special case.

### 10.2.1 Example

Lily is a swan.

```
({lily} --> swan).
```

Lily is white.

```
({lily} --> [white]).
```

Greg is a swan.

```
({greg} --> swan).
```

What color is Greg?

```
({greg} --> [?whatColor])?
```

```
A:white
Answer ({greg} --> [white]). <1,0.33>
```

## 10.3 Answering Process

Deduction and Induction is one of NARS's basic reasoning capabilities. In the former since Gertrude is a sheep and since sheeps are afraid of wolves, so is Gertrude, and in the latter: Greg is probably white because he is a swan and there is a swan, namely Lily, known which is white. Also Abduction tasks are easy for NARS, like that if it is known that Tim is white like a swan, that he might indeed be a swan.

# 11 Positional Reasoning

Many may know the SHRDLU program written by Terry Winograd [6], which is reasoning about a 3D micro domain only consisting of blocks, pyramids etc. where the user is able to ask the system questions like whether there is a red block above the green block. Also one is able to let the system perform certain actions like building new user-defined structures. From AGI perspective the reasoning capabilities which are needed for this are limited to deductions, but it is at the same time a very representative example of positional reasoning.

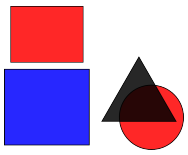
## 11.1 Example

Background knowledge about the relation between X and Y in 2-dimensional space: If A is on top of B and B is on the right of C, then A is to the right of C

```
((&&,((*,$A,#B) --> topof),((*,$C,#B) --> rightof)) ==> ((*,$C,$A) --> rightof)).
```

If A is to the right of B and B is on top of C, then A is on top of C

```
((&&,((*,$A,#B) --> rightof),((*,#B,$C) --> topof)) ==> ((*,$A,$C) --> topof)).
```



Input: The triangle is to the right of the blue square.

```
((*,triangle,(&,[blue],square)) --> rightof).
```

The red square is on top of the blue square.

```
((*,(&,[red],square),(&,[blue],square)) --> topof).
```

The red sphere is to the right of the blue square.

```
((*,(&,[red],sphere),(&,[blue],square)) --> rightof).
```

Is the red sphere to the right of the blue square?

```
((*,(&,[red],sphere),(&,[blue],square)) --> rightof)?
```

Is the red square to the left of the triangle?

```
((*,triangle,(&,[red],square)) --> rightof)?
```

A: yes

```
Answer ((*,(&,[red],sphere),(&,[blue],square)) --> rightof). <1.00,0.90>
```

A: yes

```
Answer ((*,triangle,(&,[red],square)) --> rightof). <1.00,0.15>
```

### 11.2 Answering Process

Here only deduction is needed in order to answer the yes/no questions.

## 12 Reasoning about size

### 12.1 Example

Background knowledge: Transitivity of size:

```
((&&,((*,$A,$B) --> fitsin),((*,$B,$C) --> fitsin)) ==> ((*,$A,$C) --> fitsin)).
```

Input: The football fits in the suitcase.

```
((*,{football},{suitcase}) --> fitsin).
```

The suitcase fits in the cupboard.

```
((*,{suitcase},{cupboard}) --> fitsin).
```

The box of chocolates is smaller than the football.

```
((*,{chocolatebox},{football}) --> fitsin).
```

Will the box of chocolates fit in the suitcase?

```
((*,{chocolatebox},{suitcase}) --> fitsin)?
```

A: yes

```
Answer ((*,{chocolatebox},{suitcase}) --> fitsin). <1,0.73>
```

### 12.2 Answering Process

Here only deduction is needed in order to answer the yes/no questions.

## 13 Path Finding

### 13.1 Example

Background knowledge: Path of length 2 is defined as:

```
((&&,((*,#1,#2) --> starttarget),((*,#1,#B,$C) --> positioned),
((*,#B,#2,$C2) --> positioned)) ==> ((*,id,$C,id,$C2) --> path)).

((&&,((*,#1,#2) --> starttarget),((*,#1,#B,$C) --> positioned),
((*,#2,#B,$C2) --> positioned)) ==> ((*,id,$C,neg,$C2) --> path)).

((&&,((*,#1,#2) --> starttarget),((*,#B,#1,$C) --> positioned),
((*,#B,#2,$C2) --> positioned)) ==> ((*,neg,$C,id,$C2) --> path)).
```

```
((&&,((*,#1,#2) --> starttarget),((*,#B,#1,$C) --> positioned),
((*,#2,#B,$C2) --> positioned)) ==> ((*,neg,$C,neg,$C2) --> path)).
```

Input: The kitchen is north of the hallway.

```
((*,{kitchen},{hallway},south) --> positioned).
```

The den is east of the hallway.

```
((*,{den},{hallway},west) --> positioned).
```

How do you go from den to kitchen?

```
((*,{den},{kitchen}) --> starttarget).
(?what --> path)?
```

A:west,north

```
Answer ((*,id,west,neg,south) --> path). <1.00,0.35>
```

### 13.2 Answering Process

This example demanded reasoning about a user-defined definition of a path. It could easily be extended to arbitrary path length if the related horn clause in NAL form is input, but this representation is unnatural for an AGI system, and complex pathfinding is also difficult for humans. Usually pathfinding happends implicitly, by chosing the right way to go when the situation demands it (when one reaches the crossing), instead of always having the explicit representation of the entire path in mind. If explicit large-scale-planning is needed an algorithmic planner which NARS can call by an operator may be more effective, especially if like in this domain uncertainty is not even considered.

## 14 Reasoning about Agents Motivations

### 14.1 Example

John is hungry.

```
({john} --> [hungry]). :|:
```

John goes to the kitchen.

```
({john} --> (/,go,_,{kitchen})). :|:
```

John eats the apple.

```
({john} --> (/,eat,_,{apple})). :|:
```

Daniel is hungry.

```
({daniel} --> [hungry]). :|:
```

Where does Daniel go?

```
({daniel} --> (/,go,_,{?Where}))?
```

A:kitchen

```
Answer ({daniel} --> (/,go,_,{kitchen})). <1;0.29>
```

Why did John go to the kitchen?

```
(?Why =/> ({daniel} --> (/,go,_,{kitchen})))?
```

A:hungry, was expected, but there isn't enough context to make this unambiguous, so NARS in this case thinks it is because John went to the kitchen which is also valid.

```
Answer ((&/,({john} --> (/,go,_,{kitchen})),+3) =/> ({daniel} --> (/,go,_,{kitchen}))). <1,(
```

## 14.2 Answering Process

This one involves not only the temporal induction based on

```
({john} --> [hungry]). :|:
```

and

```
({john} --> (/,go,_,{kitchen})). :|:
```

but also a variable introduction, which together leads to:

```
((&/,($1 --> [hungry]),+k) =/> ($1 --> (/,go,_,{kitchen})))
```

This is a possible base hypothesis to answer the first question, because from

```
((&/,($1 --> [hungry]),+k) =/> ($1 --> (/,go,_,{kitchen})))
```

and

```
({daniel} --> [hungry]). :|:
```

one can derive

```
({daniel} --> (/,go,_,{kitchen}). :/:
```

The answer of the system related to the why question is also valid, because there is not enough context to exclude that John going to the kitchen was also the reason why Daniel did it.

# 15 Conclusion

This paper showed how to represent the in [1, 2] proposed QA tasks in NAL so that they can be answered by NARS. We tried alle these example types with the latest stable version of the system, OpenNARS v1.6.4. If background knowledge is explicitely given to the system it is able to deal with all examples in the specific domain without further pre-training due to the semantics NAL provides. However it can not be assumed that all of these tasks will always be optimally solved by NARS due to the assumption of insufficient knowledge and resources such an attention-driven system has to make. Furthermore these examples only capture a small part of the reasoning capabilities an AGI has to have. Essential by these examples not captured parts include Uncertainty Reasoning, Introspective Inference, and Decision Making.

# References


[1] The bAbI project, https://research.facebook.com/researchers/1543934539189348

[2] Jason W., Antoine B., Sumit C., Tomas M., Alexander M. R.: Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks, arXiv:1502.05698 [cs.AI] (2015)

[3] Wang, P.: Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence. Ph.D. thesis, Indiana University (1995)

[4] Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006)

[5] Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)

[6] Terry W.: Procedures as a Representation for Data in a Computer Program for Understanding Natural Language Cognitive Psychology Vol. 3 No 1, (1972)