Dolphin

Main goals

- Knowledge Base for TIL
- Inference
- Easy data access
- World and Time support

Current achievements

- Knowledge Base without World and Time support
- Basic inference (and, or, neg, specificator)
- Inference for simple questions

Dolphin

World and Time support

- Time variable is unified with General Time object
- World variable is unified with Dolphin World
- The design makes it easy to incorporate world, time support by the time the complex inference machine is available

Inference

- program inference
- object using an INFERE program
- rule linked to the object

Dolphin

program inference

- handling variables
- performs applications
- negation and specificator

object using an INFERE program

- accessed through a standard language word in a sentence
- used only as an applicator (AND, OR, +, - …) [And ARG1 ARG2]
- contains an INFERE program (=(4 $1) =($2 4) | return(4 o) ;)

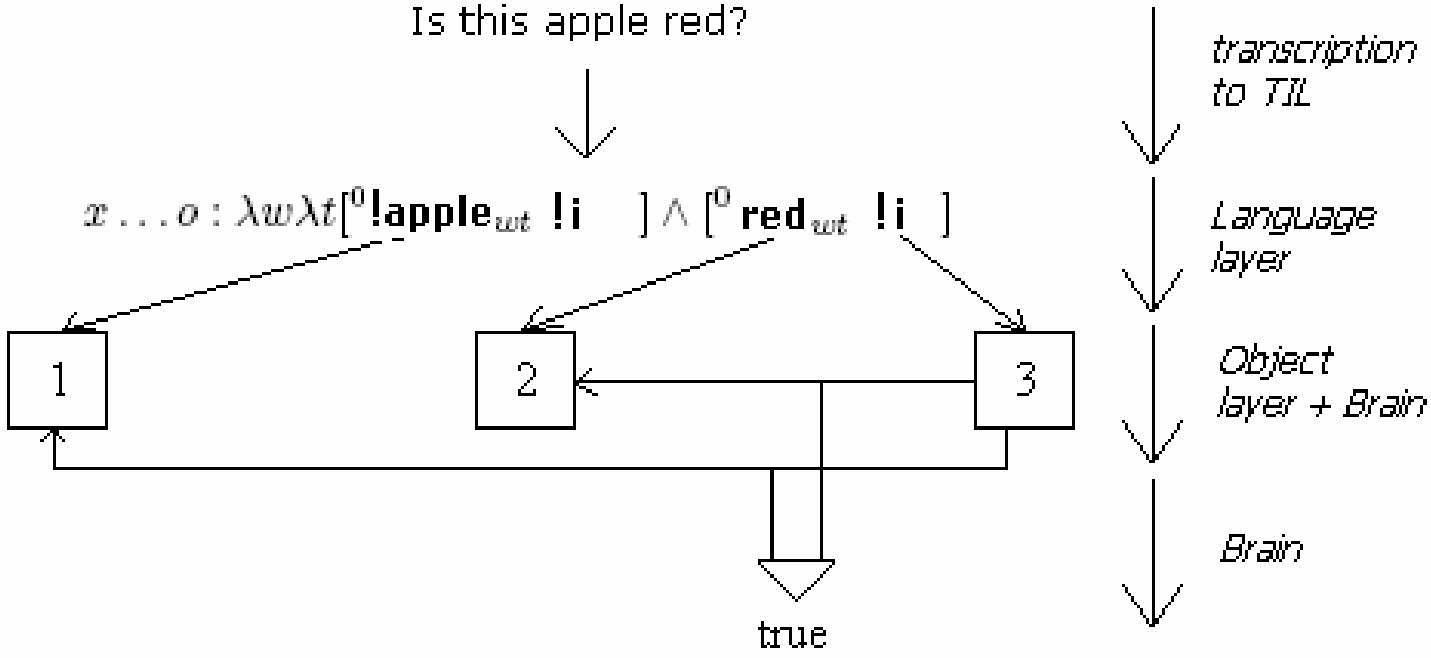
INFERE rule linked to an object

- not available yet
- if the object is used, this rule is used
- EXAMPLE: Every apple is red (INFERE program 1). This is an apple. Thanks to apple, not only facts are stored, but 1 is used too.

Dolphin

Architecture of the system

- Language Layer
- Object Layer



Dolphin

Language Layer

- Transcripts language words to objects
- Multiple language support, only one can be used
- Synonym and Homonym features, not usable yet
- All features will be available with complex inference

Object Layer

language layer

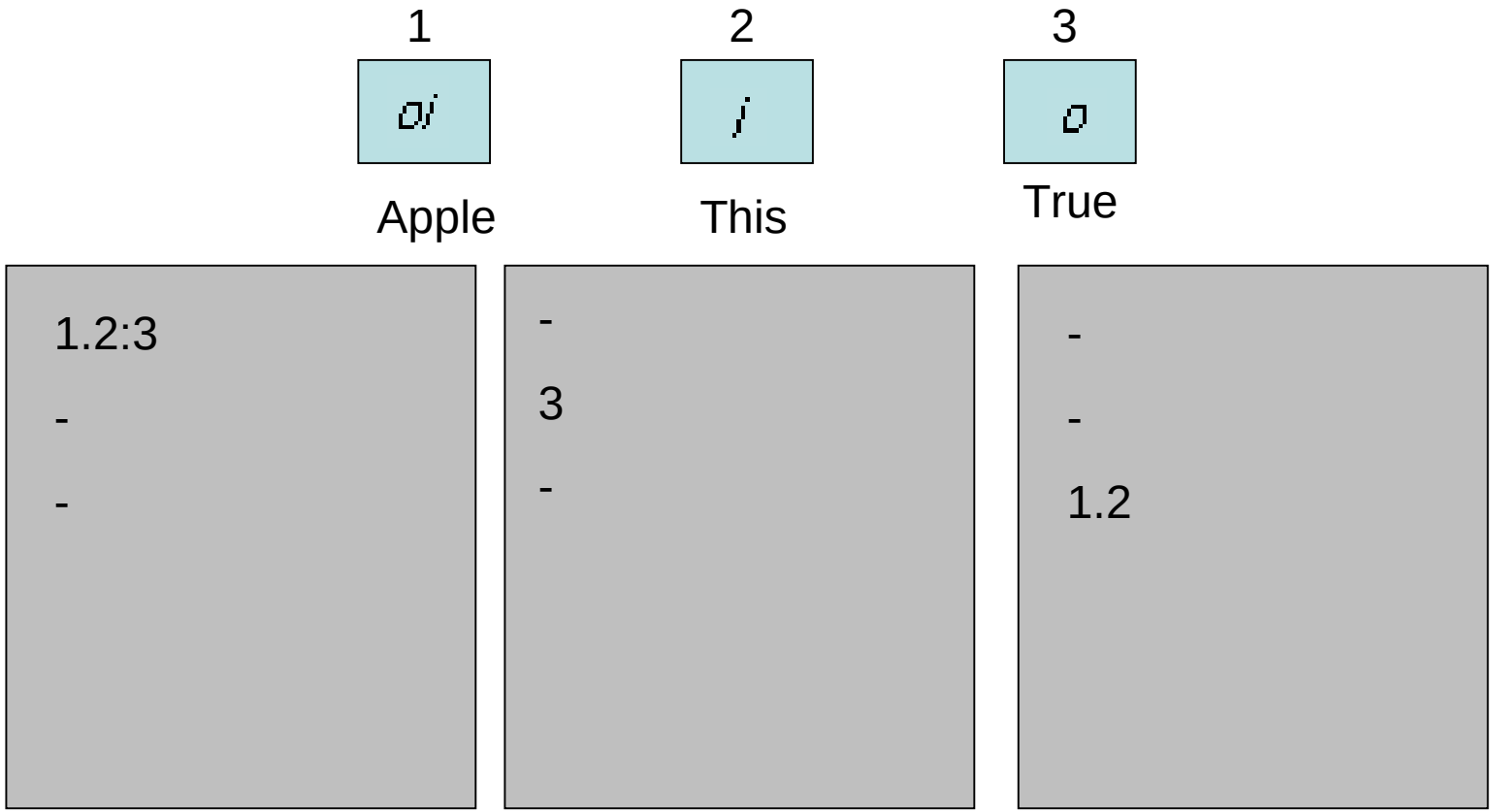
BULB

object layer

1024

Logical connection among objects

This is an apple.



Dolphin

Two modes

- Learning mode
- Question mode

Learning mode

- “.” at the end of a match
- free variables without specificator are replaced by a new object
- stores new facts
- checks basic consistence

Question mode

- “?” at the end of a match
- if an free variable occurs, *I don’t know* is raised
- simple omicron questions
- simple variable question (x…i :: red x)

Dolphin

Storage of “This is an apple.”

Til transcription is $\lambda w_1 \lambda t_2([^0\text{apple}_{w_1 t_2}, \text{This}]) \ldots \pi$

Input for Dolphin has a special format obtained from a TIL transcript of a sentence

```
3
0 w1 w
1 t2 t
2 This i
;
1
0 Apple (oi)tw
;
true...o :: lV( 0 ) lV( 1 ) [ [ [ W( 0 ) V( 0 ) ] V( 1 ) ] V( 2 ) ] .
```

Dolphin

```
true...o :: lV( 0 ) lV( 1 ) [ [ [ W( 0 ) V( 0 ) ] V( 1 ) ] V( 2 ) ] .
```

Dolphin

```
true...o :: lV( 0 ) lV( 1 ) [ [ [ W( 0 ) V( 0 ) ] V( 1 ) ] V( 2 ) ] .
```

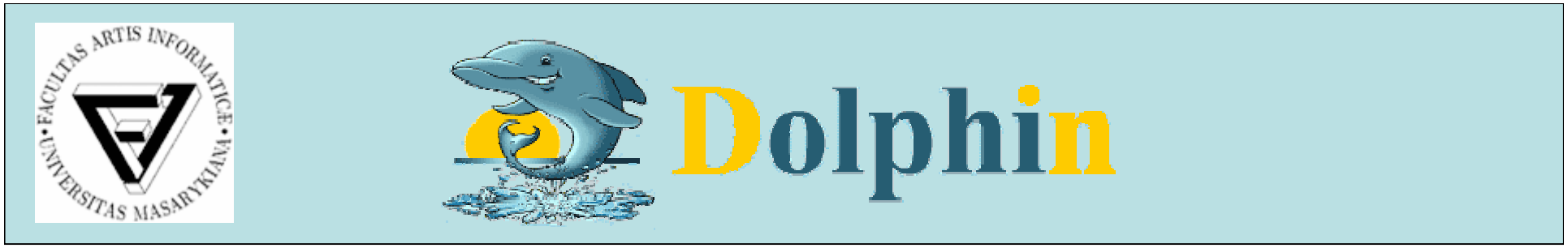
Command Prompt - Dolphin.exe -f -d -h

```
Constructions:
------------------
3. //[w-0, v-0, ] 0 0 0
2. //[c-3, v-1, ] 0 0 0
1. /0, 1, /[c-2, v-2, ] 0 0 0
HASH: string: 11.2 hash: 17d49ab14f0d4a8bbffe14ad3d6b7b13
HASH: string: 13.3 hash: dcfc3fee77e49b78055d7bef2ae2ad4f
HASH: string: 14.12 hash: fa7a332ea249be988b987dabd27bc5ac

Used variables after initialize:
---------------------------------
0 w1 w 2 0 0 0
1 t2 t 3 0 0 0
2 This i 12 0 0 0

Solved table:
-------------
1. 4 o
2. 14 (oi)
3. 13 (oi)t


Dolphin> hmm...
Please enter name of program in ./programs/:
```



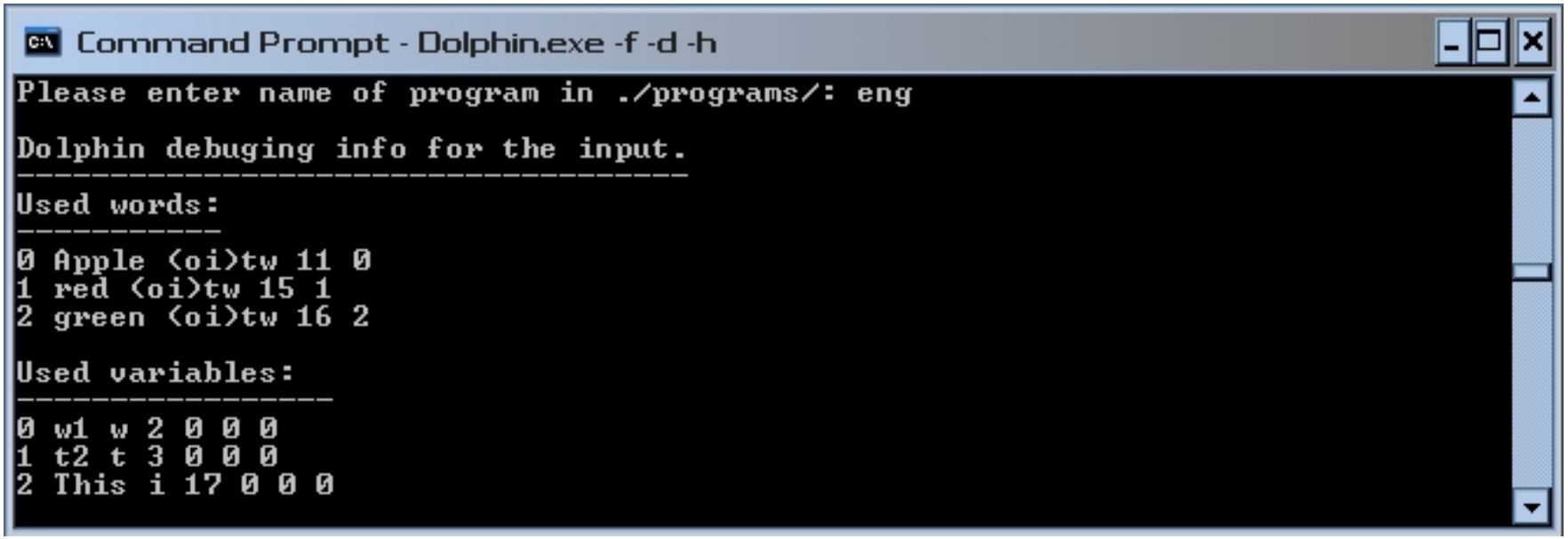
Logic OR

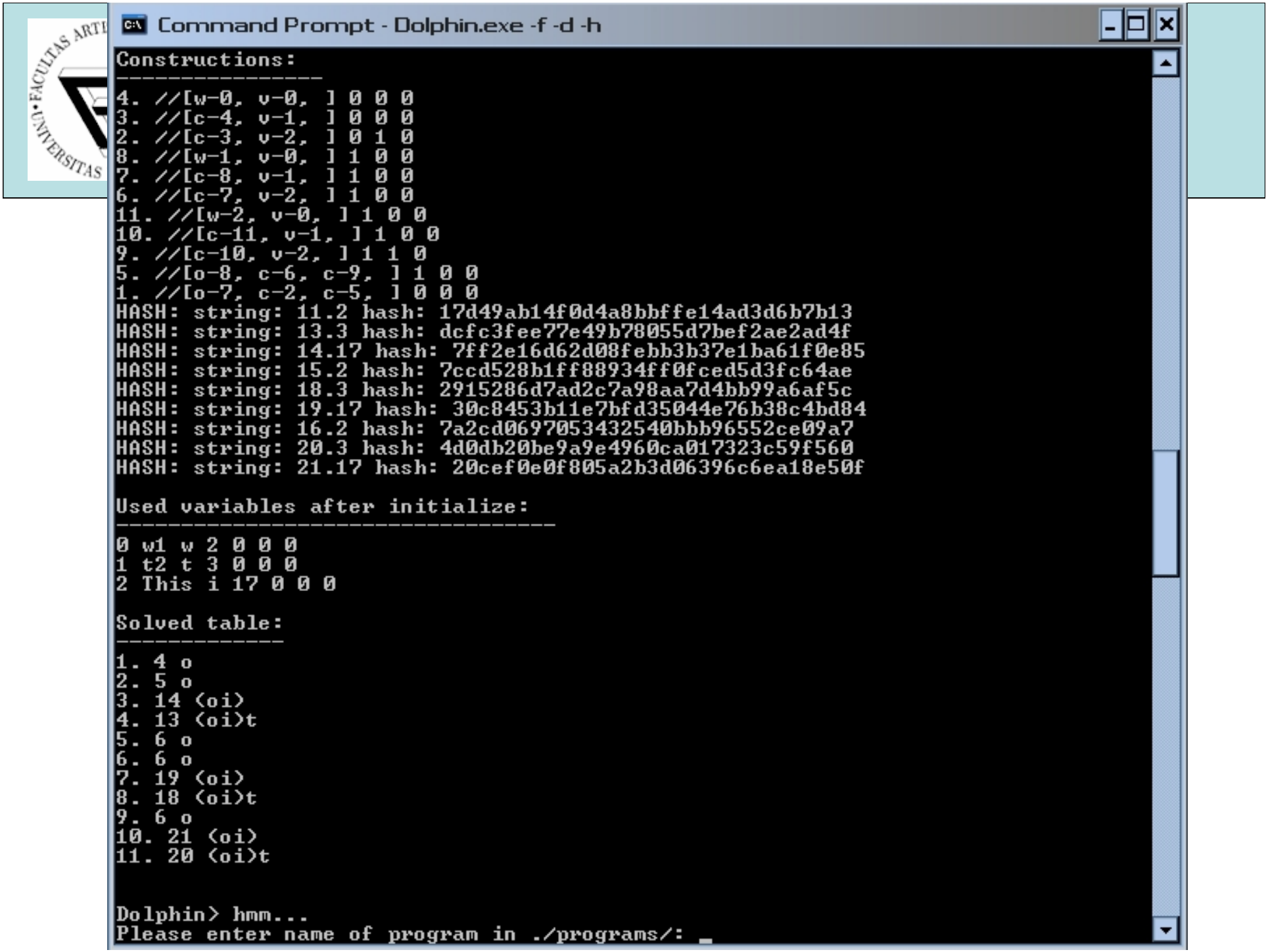
- defines new boolean variable called Both
  (This is green or red. What color it is?)

Complex example

- This is not an apple but it is red or is not green.

```
Both...o :: [ O( And ) O( Neg ) [ [ [ W( 0 ) V( 0 ) ] V( 1 ) ] V( 2 ) ]
             [ O( Or ) [ [ [ W( 1 ) V( 0 ) ] V( 1 ) ] V( 2 ) ] O( Neg ) [ [ [ W( 2 ) V( 0 ) ] V( 1 ) ] V( 2 ) ] ] ] .
```




Command Prompt - Dolphin.exe -f -d -h
Constructions:
4. //[w-0, v-0,] 0 0 0
3. //[c-4, v-1,] 0 0 0
2. //[c-3, v-2,] 0 1 0
8. //[w-1, v-0,] 1 0 0
7. //[c-8, v-1,] 1 0 0
6. //[c-7, v-2,] 1 0 0
11. //[w-2, v-0,] 1 0 0
10. //[c-11, v-1,] 1 0 0
9. //[c-10, v-2,] 1 1 0
5. //[o-8, c-6, c-9,] 1 0 0
1. //[o-7, c-2, c-5,] 0 0 0
HASH: string: 11.2 hash: 17d49ab14f0d4a8bbffe14ad3d6b7b13
HASH: string: 13.3 hash: dcfc3fee77e49b78055d7bef2ae2ad4f
HASH: string: 14.17 hash: 7ff2e16d62d08febb3b37e1ba61f0e85
HASH: string: 15.2 hash: 7ccd528b1ff88934ff0fced5d3fc64ae
HASH: string: 18.3 hash: 2915286d7ad2c7a98aa7d4bb99a6af5c
HASH: string: 19.17 hash: 30c8453b11e7bfd35044e76b38c4bd84
HASH: string: 16.2 hash: 7a2cd0697053432540bbb96552ce09a7
HASH: string: 20.3 hash: 4d0db20be9a9e4960ca017323c59f560
HASH: string: 21.17 hash: 20cef0e0f805a2b3d06396c6ea18e50f
Used variables after initialize:
0 w1 w 2 0 0 0
1 t2 t 3 0 0 0
2 This i 17 0 0 0
Solved table:
1. 4 o
2. 5 o
3. 14 (oi)
4. 13 (oi)t
5. 6 o
6. 6 o
7. 19 (oi)
8. 18 (oi)t
9. 6 o
10. 21 (oi)
11. 20 (oi)t
Dolphin> hmm...
Please enter name of program in ./programs/:

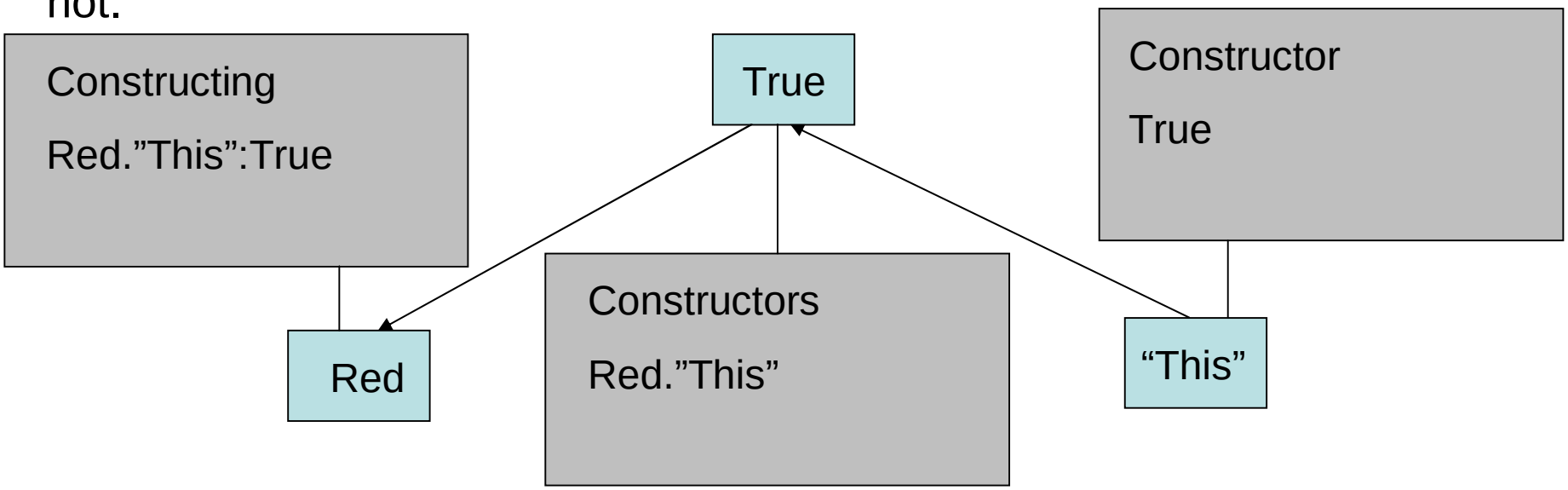
Dolphin

Questions

- Questions are handled thanks to
  1. Final product of application
  2. Logical connections between objects

1. Is Peter tall? (Peter is tall. Stored before.)
   x…o :: [0(And) [Peter x] [tall x]]
   When [Peter x] is processed, x is initialized by previously used object 12 that has connection to "Peter". After this [tall x] is processed and thanks to the Hash table true value is returned. Now we have [0(And) True True] what gives final True.
2. What red objects do you know?
   x…i :: [red x]
   Constructing section of "red" has information what is red and not. Record 11.28:4 tells us that object 28 is "red" (11). Thanks to this, it is enough to search in this section by using 11.0:4 where 0 represents variable. This produces front of possible values of x that is than printed on the standard output.

Question means to identify suitable section of an object that is used for search. Generally we look in some constructing section to find out whether objects construct another objects.
Thanks to the design of the database, it will be also possible to ask “What do you know about “this”?”. In this case, system takes Constructor section of “this” and looks what objects where created using it. If it is omicron object another search is done to find out what prorerty “this” has or not.



Dolphin

Specificator

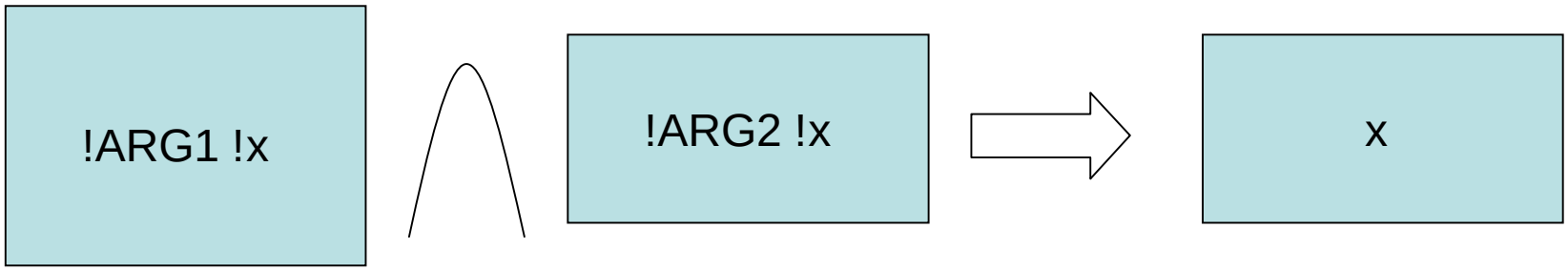
- makes it possible to use already mentioned object.
  1. This cube is red.

$$\lambda w_1 \lambda t_2([^0! \text{ cube } w_1 t_2, i_3] \wedge [^0 \text{red } w_1 t_2, i_3]) \ldots \pi$$

  2. Instead of initialization of variable i3 with a new object, previously stored object is searched
  3. Continues in standard way
- Searches for suitable objects in a way similar to the question mode
- If more suitable objects are founded, before storage of new fact user is requested to choose one particular object.
- Use of specifiator with AND, OR logic operands complicate the situation

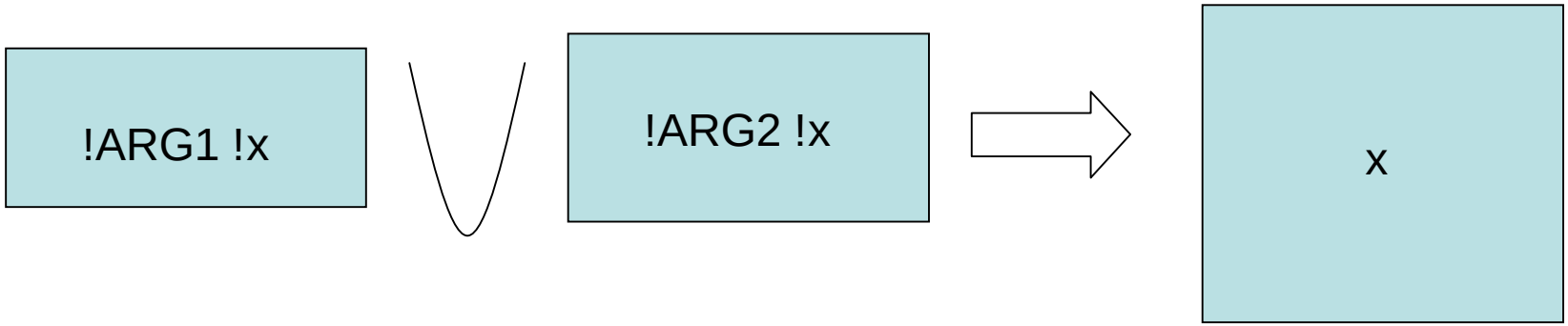
Specificator and AND

[0(And) !ARG1 !x !ARG2 !x]



Specificator and OR

[0(Or) !ARG1 !x !ARG2 !x]



Dolphin

System implementation

- Implemented in C language
- compatible with Windows and Linux
- uses B trees for fast searching in stored facts

To do

- Translator for Synt
- Time and World support
- Complex inference