SAT Planning in Description Logics:

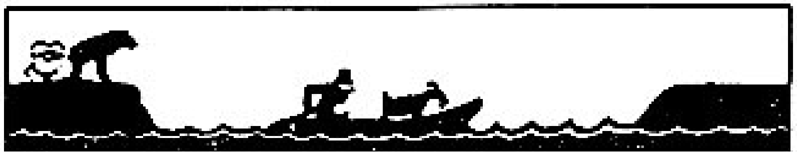
Solving the Classical “Wolf Goat Cabbage” Riddle

Michael Wessel

2014 – 06 - 30

Wolf – Goat (Sheep) – Cabbage Riddle

A



A shepherd (= “ferryman” in the following), wolf, goat, and cabbage want to cross the river.

The boat only hosts two.

The wolf and goat (sheep?) cannot be left alone together.

The cabbage and goat cannot be left alone.

How can they safely cross the river?

?

B

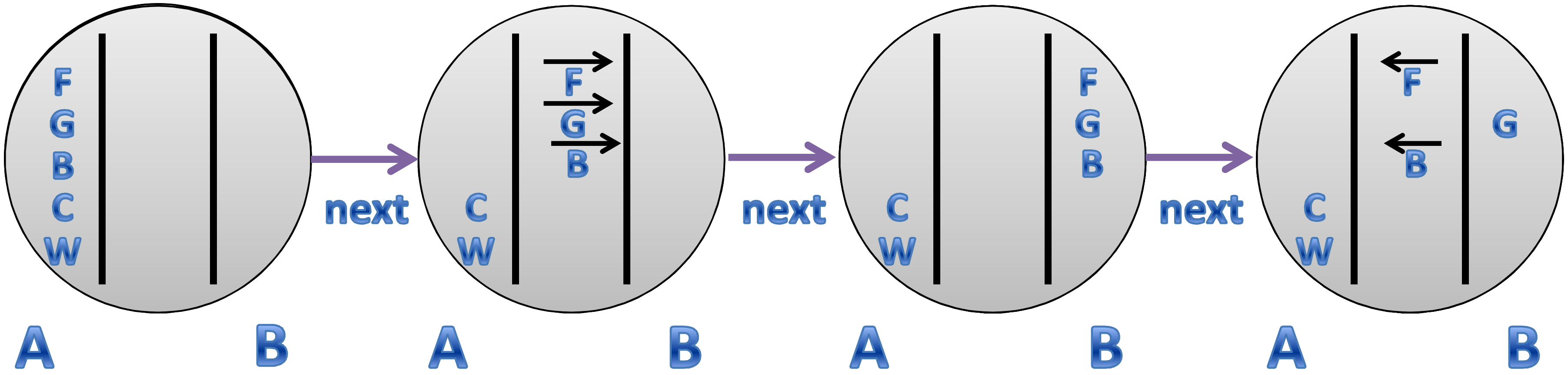
Sat Plan

- Is a planning problem
- Reduction of planning to SAT (propositional case)
  - **SAT (plan /\ start /\ goal /\ actions /\ …) = true**
    iff
    Plan = <step_1 = start, step_2, …, step_n = goal>

    (see Russell & Norvig's **AIMA book** for full details)
  - Problem: length n of plan unknown (try all…)
  - Propositional logic: proliferation of symbols
- Here – more "symbol" efficient reduction to modal logic / description logics

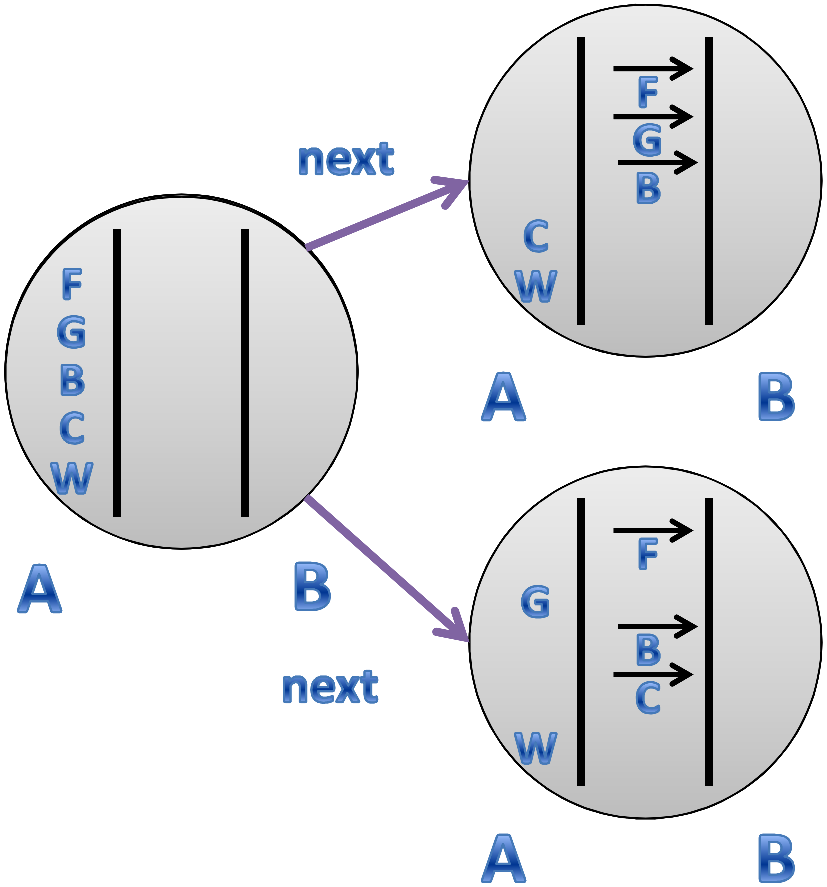
Logical Signature

- Next is a functional relation (feature)
- Each "logical symbol" can be in 4 different states
  - goat-A (goat on A river bank)
  - goat-B (goat on B river bank)
  - goat-on-boat-from-A-to-B
  - goat-on-boat-from-B-to-A



Axioms to Constrain Possible Worlds

- Need axioms that describe the possible next states



Successor State Axioms

- If the boat is on riverbank A, the next it is going from A to B
- If the boat is going from A to B, then next it is on riverbank B, etc.

```
(implies boat-a
         (all next boat-from-a-to-b))

(implies boat-from-a-to-b
         (all next boat-b))

(implies boat-b
         (all next boat-from-b-to-a))

(implies boat-from-b-to-a
         (all next boat-a))
```

$$\forall x : \text{boat-from-a-to-b}(x) \Rightarrow \\ \forall y : \text{next}(x, y) \Rightarrow \text{boat-b}(y)$$

KRSS Syntax

Some More Constraints

- If the boat is going from A to B, then
  - the boat cannot go alone (the ferryman has to go with it)
  - It should not go with only the ferryman – however, it can go back from B to A with only the ferryman!
  - the boat has capacity for ferryman and one other object

```
(implies boat-from-a-to-b
         (and ferryman-boat-from-a-to-b
                (or cabbage-boat-from-a-to-b
                    wolf-boat-from-a-to-b
                    goat-boat-from-a-to-b)
                (not (and cabbage-boat-from-a-to-b
                          wolf-boat-from-a-to-b))
                (not (and cabbage-boat-from-a-to-b
                          goat-boat-from-a-to-b))
                (not (and wolf-boat-from-a-to-b
                          goat-boat-from-a-to-b))))
```

$$\begin{aligned}
&\forall x : \text{boat-from-a-to-b}(x) \Rightarrow \\
&\quad \text{ferryman-boat-from-a-to-b}(x) \wedge \\
&\quad (\, \text{cabbage-boat-from-a-to-b}(x) \vee \\
&\quad \text{wolf-boat-from-a-to-b}(x) \vee \\
&\quad \text{goat-boat-from-a-to-b}(x) \,) \wedge \\
&\quad \sim (\, \text{cabbage-boat-from-a-to-b}(x) \wedge \\
&\quad \text{wolf-boat-from-a-to-b}(x) \,) \wedge \\
&\quad \sim (\, \text{cabbage-boat-from-a-to-b}(x) \wedge \\
&\quad \text{goat-boat-from-a-to-b}(x) \,) \wedge \\
&\quad \sim (\, \text{wolf-boat-from-a-to-b}(x) \wedge \\
&\quad \text{goat-boat-from-a-to-b}(x) \,)
\end{aligned}$$

Further Axioms

- Ensure cabbage and goat, and wolf and goat, are not alone
- Ensure every object can only be at one place at a time
- Make sure objects don't disappear – every "state" specifies that goat, boat, etc. exists "somewhere"

```
(implies (and wolf-a goat-a) ferryman-a)
(implies (and wolf-b goat-b) ferryman-b)

(implies (and cabbage-a goat-a) ferryman-a)
(implies (and cabbage-b goat-b) ferryman-b)

(disjoint goat-a goat-b goat-boat-from-a-to-b goat-boat-from-b-to-a)

(implies goat
  (or goat-a goat-b goat-boat-from-a-to-b goat-boat-from-b-to-a))
```

States, Start, Goal

- Every state is either a goal or has some next state
- Every state specifies the state of all objects
- Start state = all objects on riverbank A
- Goal state = all objects on riverbank B

```
(implies goat
         (or goat-a goat-b goat-boat-from-a-to-b goat-boat-from-b-to-a))

(implies state
         (and (or goal (some next state))
               goat wolf cabbage ferryman boat))

(implies start
         (and state boat-a goat-a wolf-a cabbage-a ferryman-a))

(implies goal
         (and state boat-b goat-b wolf-b cabbage-b ferryman-b))
```

KRSS Syntax

Verifying a “Solution”

- Check satisfiability of formula

```
(and start
        (some next
                (some next
                      (some next                         n times
                               …
                                    (some next goal))))
```

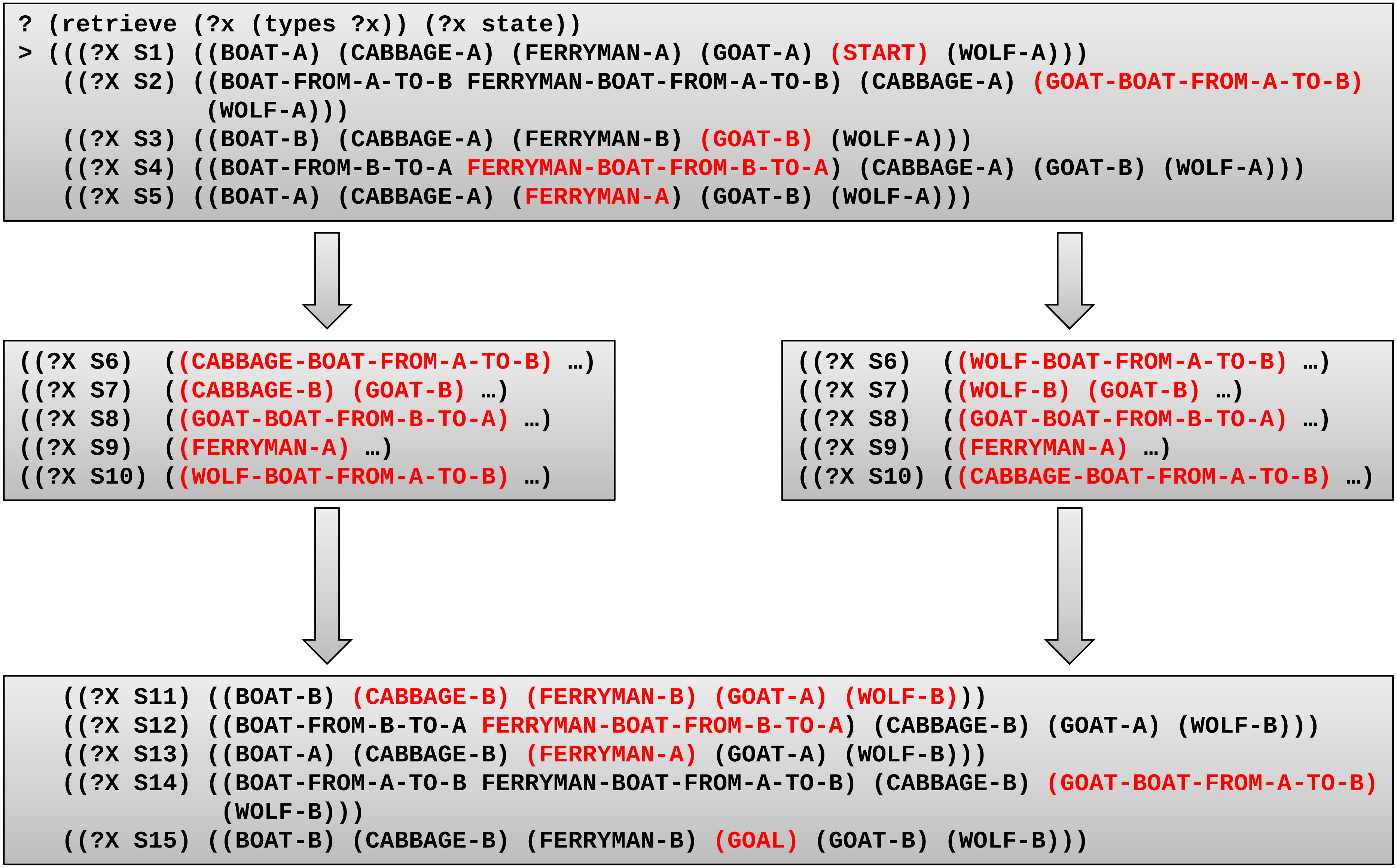
- There is a solution for n = 14
  - and also a surprising one
  - however, we need an ABox to read off the solution!

Create ABox to Read off Entailed Individual Types via Abox Query

```
(instance s1 start)
(related s1 s2 next)
(related s2 s3 next)
(related s3 s4 next)
        …
(related s13 s14 next)
(related s14 s15 next)
(instance s15 goal)
```

```
? (retrieve (?x (types ?x)) (?x state))
> (((?X S1)  ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-A) (START) (WOLF-A)))
   ((?X S2)  ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (CABBAGE-A) (GOAT-BOAT-FROM-A-TO-B)
                  (WOLF-A)))
   ((?X S3)  ((BOAT-B) (CABBAGE-A) (FERRYMAN-B) (GOAT-B) (WOLF-A)))
   ((?X S4)  ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-A) (GOAT-B) (WOLF-A)))
   ((?X S5)  ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-B) (WOLF-A)))
   ((?X S6)  ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (GOAT-B) ))
   ((?X S7)  ((BOAT-B) (FERRYMAN-B) (GOAT-B) ))
   ((?X S8)  ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (GOAT-BOAT-FROM-B-TO-A) ))
   ((?X S9)  ((BOAT-A) (FERRYMAN-A) (GOAT-A) ))
   ((?X S10) ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (GOAT-A) ))
   ((?X S11) ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAT-A) (WOLF-B)))
   ((?X S12) ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-B) (GOAT-A) (WOLF-B)))
   ((?X S13) ((BOAT-A) (CABBAGE-B) (FERRYMAN-A) (GOAT-A) (WOLF-B)))
   ((?X S14) ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (CABBAGE-B) (GOAT-BOAT-FROM-A-TO-B)
                 (WOLF-B)))
   ((?X S15) ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAL) (GOAT-B) (WOLF-B)))
```

Two Possible Solutions!



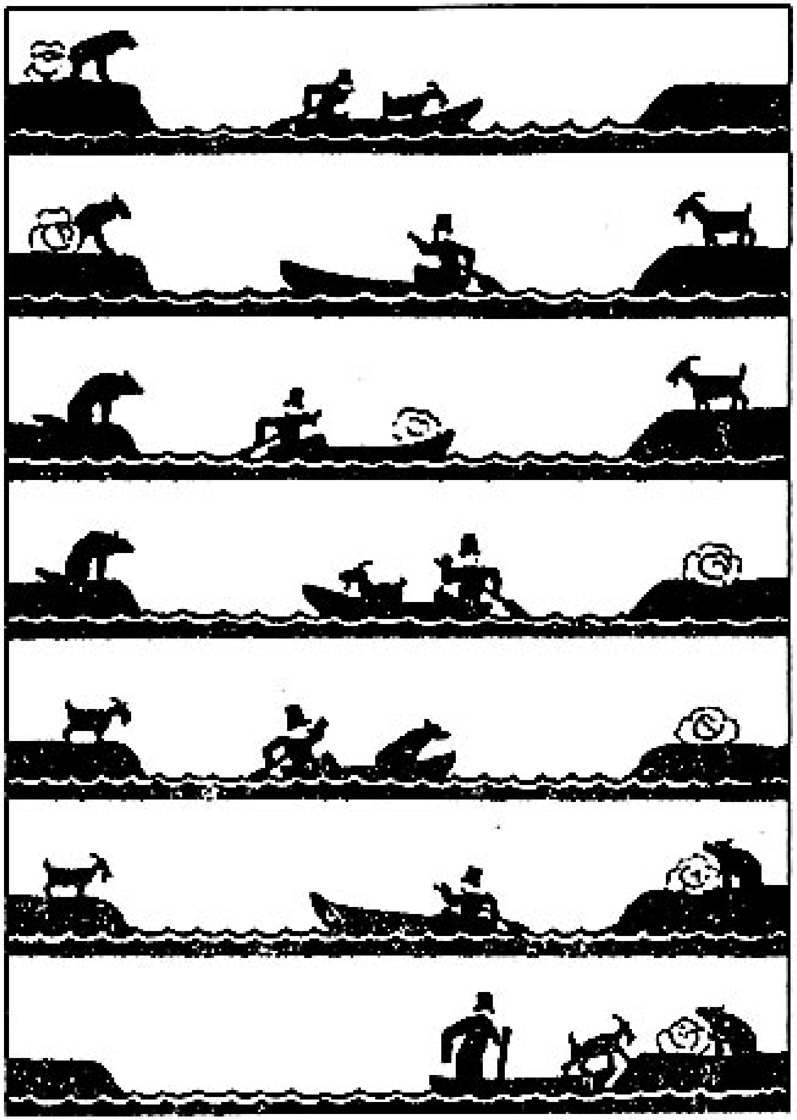
```
? (retrieve (?x (types ?x)) (?x state))
> (((?X S1) ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-A) (START) (WOLF-A)))
   ((?X S2) ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (CABBAGE-A) (GOAT-BOAT-FROM-A-TO-B)
             (WOLF-A)))
   ((?X S3) ((BOAT-B) (CABBAGE-A) (FERRYMAN-B) (GOAT-B) (WOLF-A)))
   ((?X S4) ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-A) (GOAT-B) (WOLF-A)))
   ((?X S5) ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-B) (WOLF-A)))
```

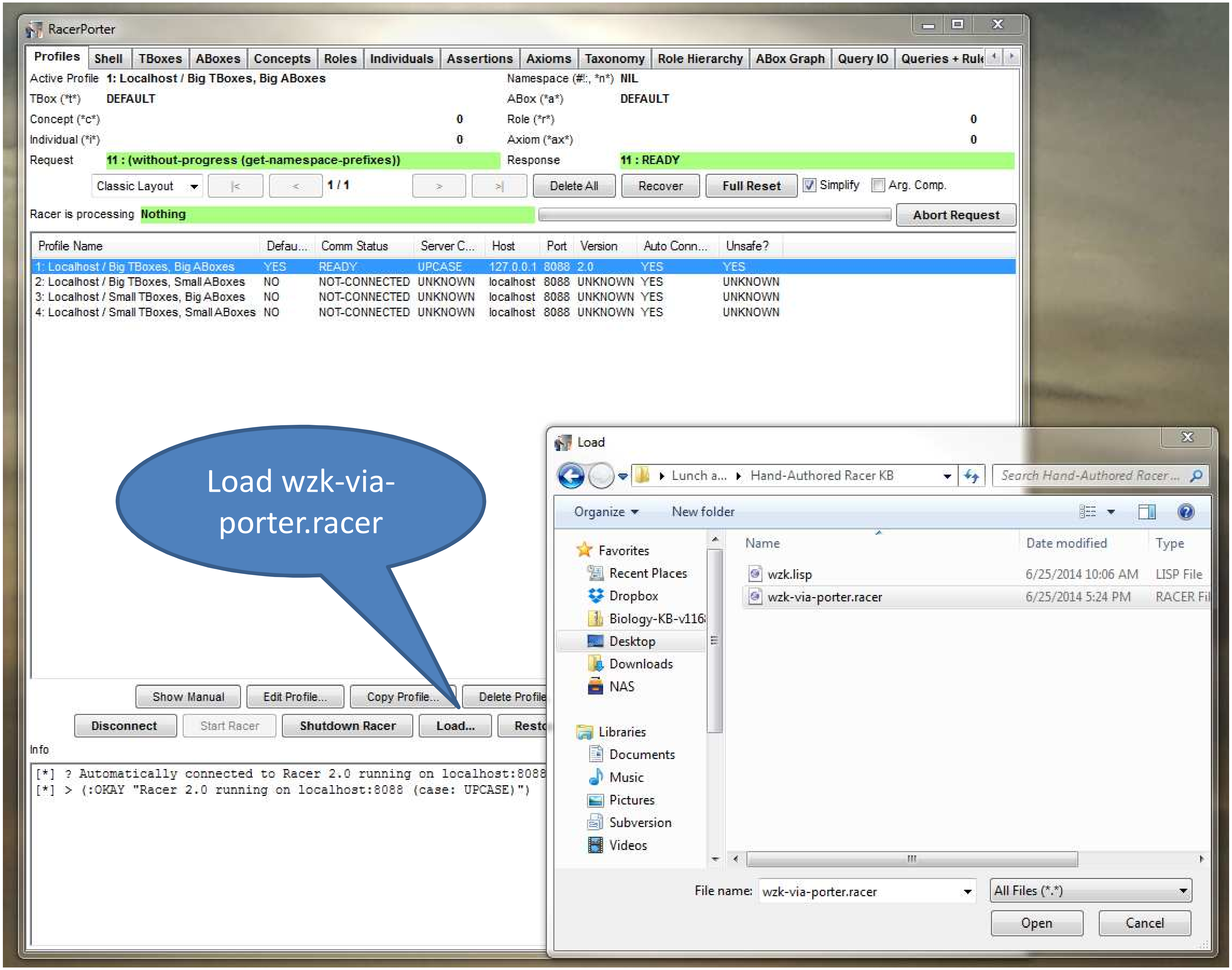
```
((?X S6)  ((CABBAGE-BOAT-FROM-A-TO-B) …)
((?X S7)  ((CABBAGE-B) (GOAT-B) …)
((?X S8)  ((GOAT-BOAT-FROM-B-TO-A) …)
((?X S9)  ((FERRYMAN-A) …)
((?X S10) ((WOLF-BOAT-FROM-A-TO-B) …)
```

```
((?X S6)  ((WOLF-BOAT-FROM-A-TO-B) …)
((?X S7)  ((WOLF-B) (GOAT-B) …)
((?X S8)  ((GOAT-BOAT-FROM-B-TO-A) …)
((?X S9)  ((FERRYMAN-A) …)
((?X S10) ((CABBAGE-BOAT-FROM-A-TO-B) …)
```

```
   ((?X S11) ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAT-A) (WOLF-B)))
   ((?X S12) ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-B) (GOAT-A) (WOLF-B)))
   ((?X S13) ((BOAT-A) (CABBAGE-B) (FERRYMAN-A) (GOAT-A) (WOLF-B)))
   ((?X S14) ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (CABBAGE-B) (GOAT-BOAT-FROM-A-TO-B)
              (WOLF-B)))
   ((?X S15) ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAL) (GOAT-B) (WOLF-B)))
```

The Official Solution





Load wzk-via-porter.racer

Active Profile 1: Localhost / Big TBoxes, Big ABoxes

Namespace (#!:, *n*) NIL

TBox (*t*) DEFAULT

ABox (*a*) DEFAULT

Concept (*c*) 0

Role (*r*) 0

Individual (*i*) 0

Axiom (*ax*) 0

Request 11 : (without-progress (get-namespace-prefixes))

Response 11 : READY

☑ Simplify ☐ Arg. Comp.

Racer is processing Nothing

Profile Name	Defau...	Comm Status	Server C...	Host	Port	Version	Auto Conn...	Unsafe?
1: Localhost / Big TBoxes, Big ABoxes	YES	READY	UPCASE	127.0.0.1	8088	2.0	YES	YES
2: Localhost / Big TBoxes, Small ABoxes	NO	NOT-CONNECTED	UNKNOWN	localhost	8088	UNKNOWN	YES	UNKNOWN
3: Localhost / Small TBoxes, Big ABoxes	NO	NOT-CONNECTED	UNKNOWN	localhost	8088	UNKNOWN	YES	UNKNOWN
4: Localhost / Small TBoxes, Small ABoxes	NO	NOT-CONNECTED	UNKNOWN	localhost	8088	UNKNOWN	YES	UNKNOWN

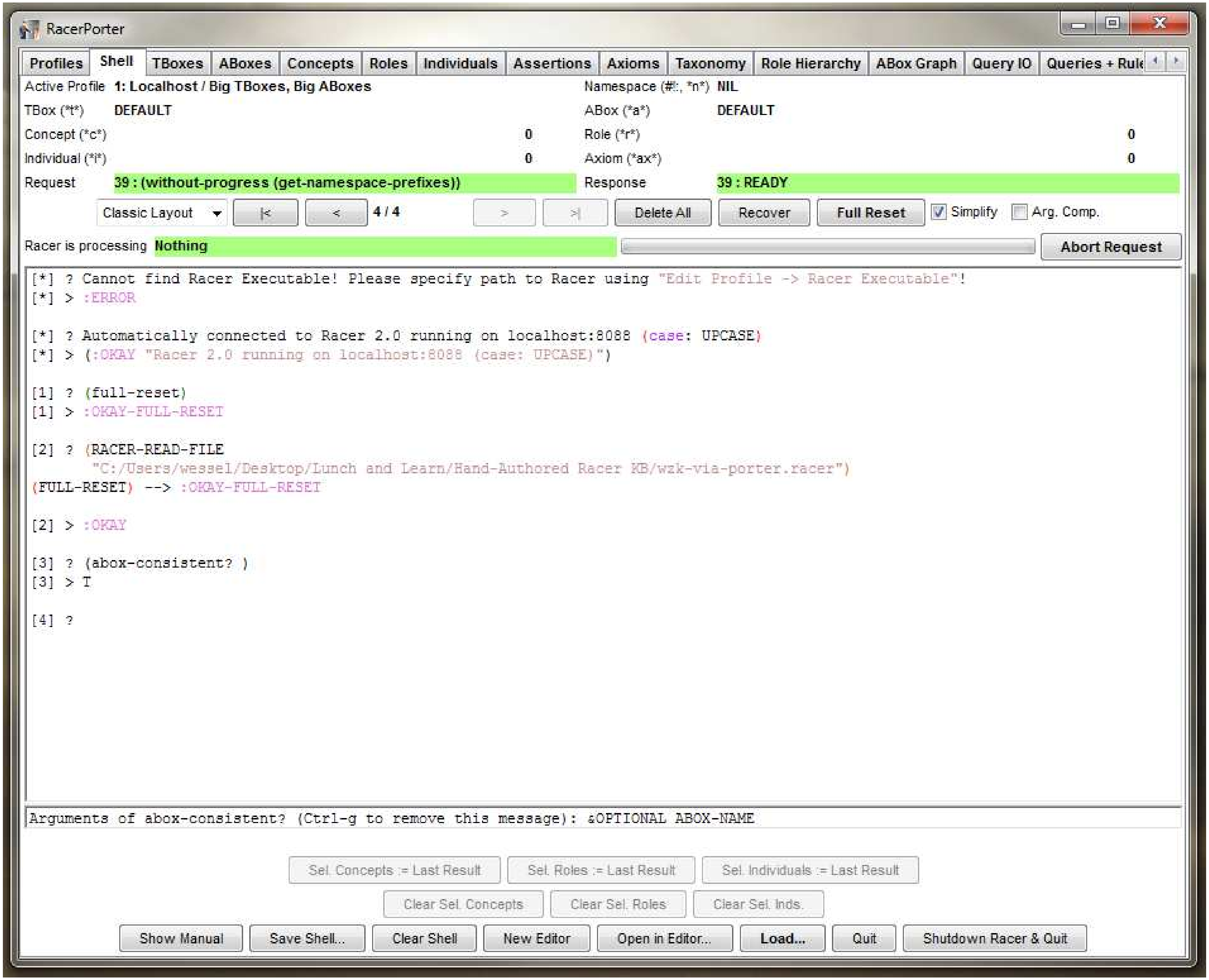
Info

```
[*] ? Automatically connected to Racer 2.0 running on localhost:8088
[*] > (:OKAY "Racer 2.0 running on localhost:8088 (case: UPCASE)")
```

Name	Date modified	Type
wzk.lisp	6/25/2014 10:06 AM	LISP File
wzk-via-porter.racer	6/25/2014 5:24 PM	RACER Fil

File name: wzk-via-porter.racer

All Files (*.*)



Active Profile 1: Localhost / Big TBoxes, Big ABoxes
Namespace (#!:, *n*) NIL
TBox (*t*) DEFAULT
ABox (*a*) DEFAULT
Concept (*c*) 0
Role (*r*) 0
Individual (*i*) 0
Axiom (*ax*) 0
Request 39 : (without-progress (get-namespace-prefixes))
Response 39 : READY

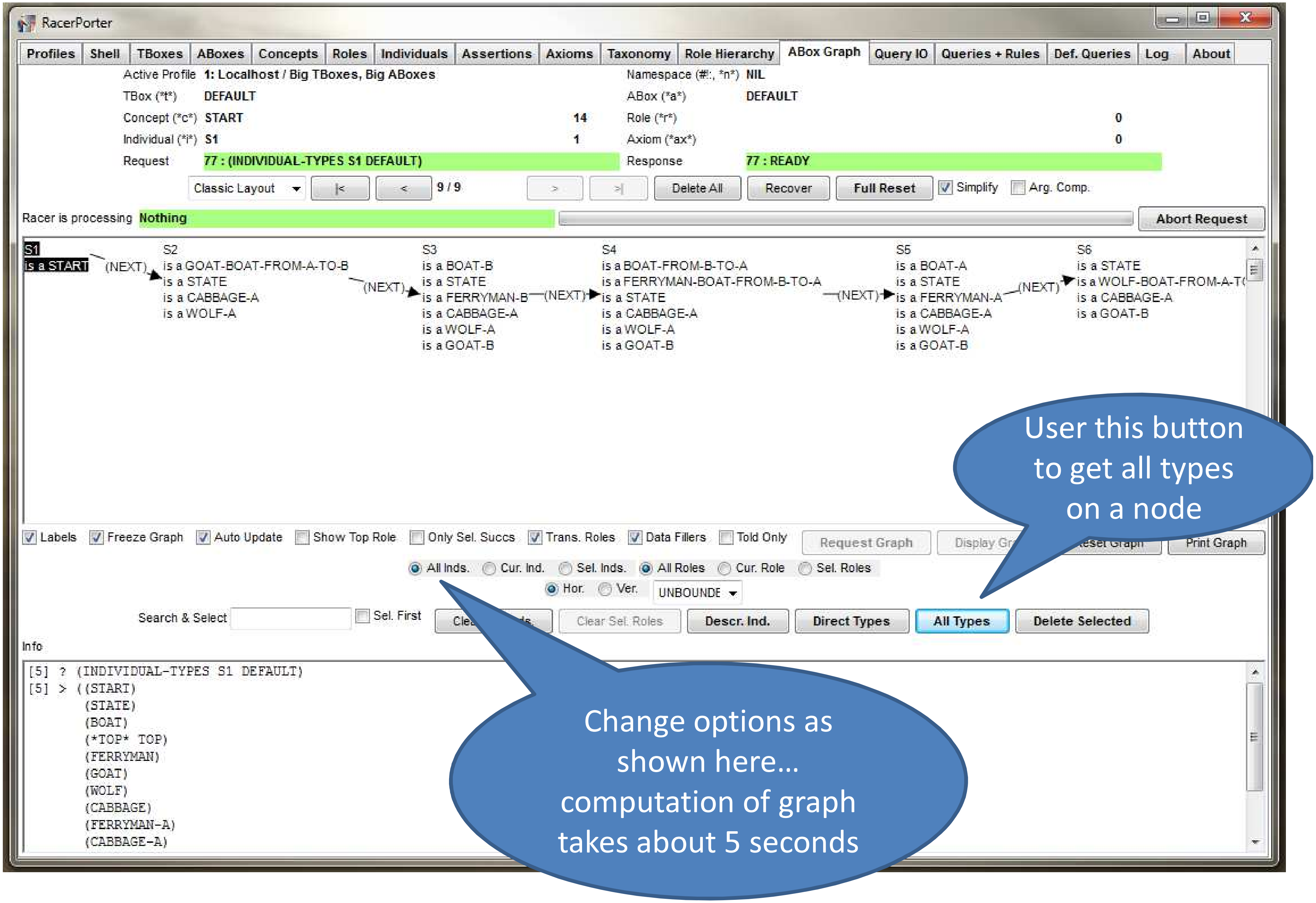
4 / 4

☑ Simplify ☐ Arg. Comp.

Racer is processing Nothing

```
[*] ? Cannot find Racer Executable! Please specify path to Racer using "Edit Profile -> Racer Executable"!
[*] > :ERROR

[*] ? Automatically connected to Racer 2.0 running on localhost:8088 (case: UPCASE)
[*] > (:OKAY "Racer 2.0 running on localhost:8088 (case: UPCASE)")

[1] ? (full-reset)
[1] > :OKAY-FULL-RESET

[2] ? (RACER-READ-FILE
       "C:/Users/wessel/Desktop/Lunch and Learn/Hand-Authored Racer KB/wzk-via-porter.racer")
(FULL-RESET) --> :OKAY-FULL-RESET

[2] > :OKAY

[3] ? (abox-consistent? )
[3] > T

[4] ?
```

```
Arguments of abox-consistent? (Ctrl-g to remove this message): &OPTIONAL ABOX-NAME
```


RacerPorter
Profiles | Shell | TBoxes | ABoxes | Concepts | Roles | Individuals | Assertions | Axioms | Taxonomy | Role Hierarchy | ABox Graph | Query IO | Queries + Rules | Def. Queries | Log | About

Active Profile 1: Localhost / Big TBoxes, Big ABoxes
Namespace (#!:, *n*) NIL
TBox (*t*) DEFAULT
ABox (*a*) DEFAULT
Concept (*c*) START 14
Role (*r*) 0
Individual (*i*) S1 1
Axiom (*ax*) 0
Request 77 : (INDIVIDUAL-TYPES S1 DEFAULT)
Response 77 : READY

☑ Simplify ☐ Arg. Comp.

Racer is processing Nothing

☑ Labels ☑ Freeze Graph ☑ Auto Update ☐ Show Top Role ☐ Only Sel. Succs ☑ Trans. Roles ☑ Data Fillers ☐ Told Only

● All Inds. ○ Cur. Ind. ○ Sel. Inds. ● All Roles ○ Cur. Role ○ Sel. Roles

● Hor. ○ Ver. UNBOUNDE

Search & Select ☐ Sel. First

Info

```
[5] ? (INDIVIDUAL-TYPES S1 DEFAULT)
[5] > ((START)
       (STATE)
       (BOAT)
       (*TOP* TOP)
       (FERRYMAN)
       (GOAT)
       (WOLF)
       (CABBAGE)
       (FERRYMAN-A)
       (CABBAGE-A)
```

User this button to get all types on a node

Change options as shown here… computation of graph takes about 5 seconds