On the Use of Computational Argumentation for Real World Applications

Pavlos Moraitis

LIPADE

Paris Descartes University

Seminar@SystemX

24 JANVIER 2019

Overview

- Introduction
- Basics on Argumentation
- CAFs and application to autonomous driving
- LPP-GORGIAS and some applications
- Conclusions

Why Argumentation? Human like Systems

- **Natural Intelligence or high-level cognition is manifested by its handling of Conflicting Information**

- **Argumentation is native to human reasoning**
  - Role of argumentation in natural human reasoning and dialogue studied in philosophy, linguistics, psychology, …

- **Knowledge captured as arguments**

- **Aristotle: "Dialectic Argument" for handling conflicting positions/claims**

Why Argumentation? Logical Reasoning

- **Formal Logic in terms of Argumentation**
  - **Argumentation unifies strict/formal and informal reasoning**

- **Argumentation is the primary notion of reasoning.**

What is Computational Argumentation?

- Argumentation can be abstractly defined as the formal interaction of different conflicting arguments for and against some claim
  - arguments = proofs of claims in some underlying logic
  - The claims may represent beliefs, goals and actions

- Argumentation process
  - Construction of arguments (based on different underlying monotonic logics)
  - Definition of interactions between arguments (based on different notions of conflicts)
  - Evaluation of strength of arguments (by using preferences, values, etc.)
  - Definition of status of arguments (i.e. accepted, rejected, undecided based on different acceptability semantics)
  - Choice of winning arguments

What is an argument?

- **A set of premises supporting a conclusion /claim**

the claim

A1

Information INFO about Paul should be published

because

the premises

Paul has political responsibilities

and

INFO is in the national interest

and

if a person has political responsibilities and info about that person is in the national interest then that info should be published

So argumentation is…

- The process whereby arguments are constructed, (possibly) exchanged and evaluated in light of their interactions with other arguments.

  - √ X √ A1: (publish info about Paul because he has political responsibilities)
  - √ X A2: (Paul does not have political responsibilities because he resigned)
  - √ A3 : (Paul has still political responsibilities because his resignation has not been accepted)

Argumentation Logics

- **Argumentation logics** formalise defeasible reasoning as construction and comparison of arguments

- Use of monotonic logics for modeling **non-monotonic reasoning** based on the interaction of arguments in the presence of **uncertain**, **incomplete** and **conflicting** knowledge/information

Abstract Argumentation Frameworks

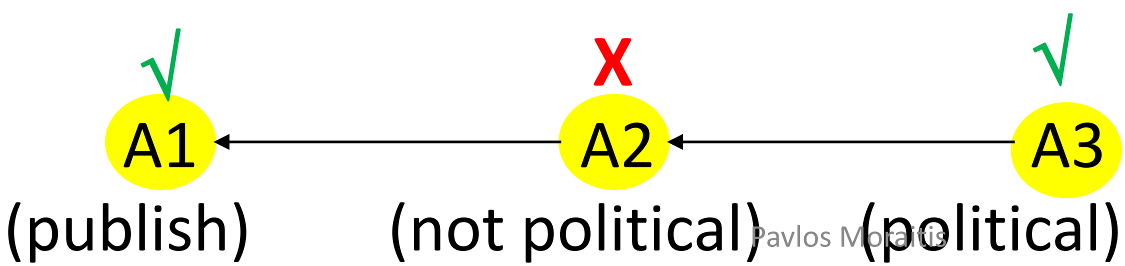
- **Structure** of arguments is not specified
- **Semantics** help us to choose the « good/winning » arguments
- Dung's [Dung 95] **acceptability semantics** for abstract argumentation frameworks

Dung's Abstract Argumentation Framework

- AF=<Args, Attack> where
  - Args={a1,…,an} is a set of arguments
  - Attack $\subseteq$ A x A is a binary attacking relation

- (Args, Attack) abstracts from underlying logic based definition of Args and Attack

- Application of semantics allows us determine winning arguments

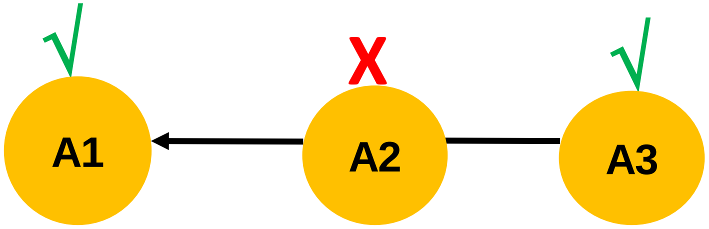


Extension-based Semantics

- **Defense: counter-attacking of all received attacks**

- **(Admissible) Extension $\mathcal{E}$: conflict-free set of arguments that defends all its members**

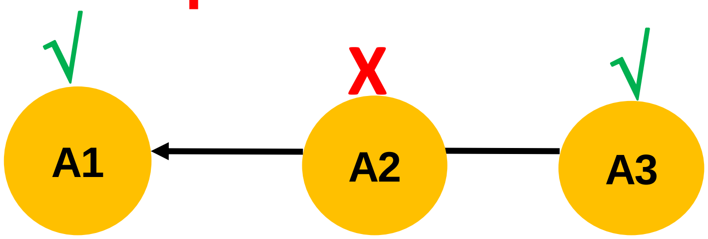
- **Skeptical and credulous acceptability**



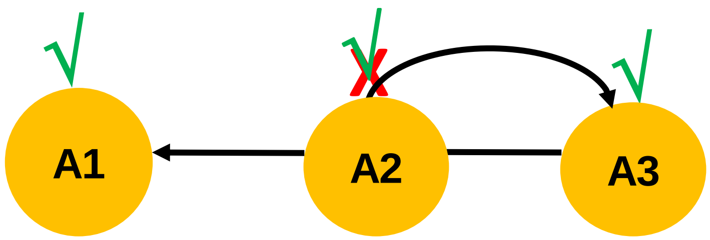
$\mathcal{E}$={A1,A3}

Extension-based Semantics

- **Defense: counter-attacking of all received attacks**
- **(Admissible) Extension $\mathcal{E}$: conflict-free set of arguments that defends all its members**
- **Skeptical and credulous acceptability**



$\mathcal{E}$={A1,A3}



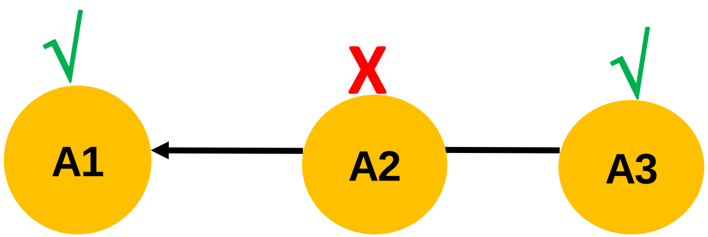
$\mathcal{E}_1$={A1,A3}, $\mathcal{E}_2$={A2}

Extension-based Semantics

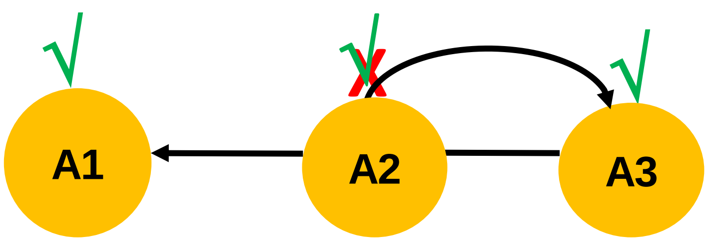
- **Defense: counter-attacking of all received attacks**

- **(Admissible) Extension $\mathcal{E}$: conflict-free set of arguments that defends all its members**

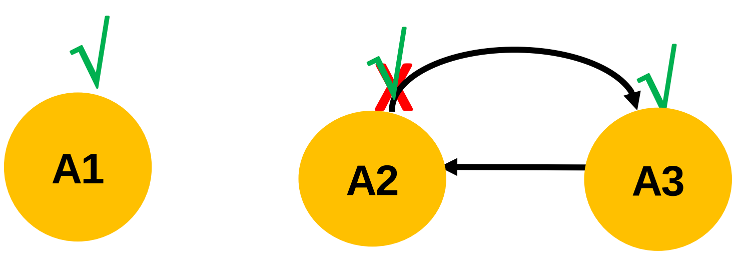
- **Skeptical and credulous acceptability**



$\mathcal{E}$={A1,A3}



$\mathcal{E}_1$={A1,A3}, $\mathcal{E}_2$={A2}



$\mathcal{E}_1$={A1,A3}, $\mathcal{E}_2$={A1,A2}

Dynamics in Argumentation

- **Addition/Removal of Arguments**
- **Addition/Removal of Attacks**
- **Goal: obtaining the acceptance of a particular (set of) argument(s)**

Control Argumentation Frameworks (CAFs)
[Dimopoulos, Mailly, Moraitis (AAAI18)]

- **A CAF is an argumentation framework where arguments are divided in three parts: fixed, uncertain and control**

- **Fixed: background knowledge about a static environment**

- **Uncertain: changes that may occur in the environment**

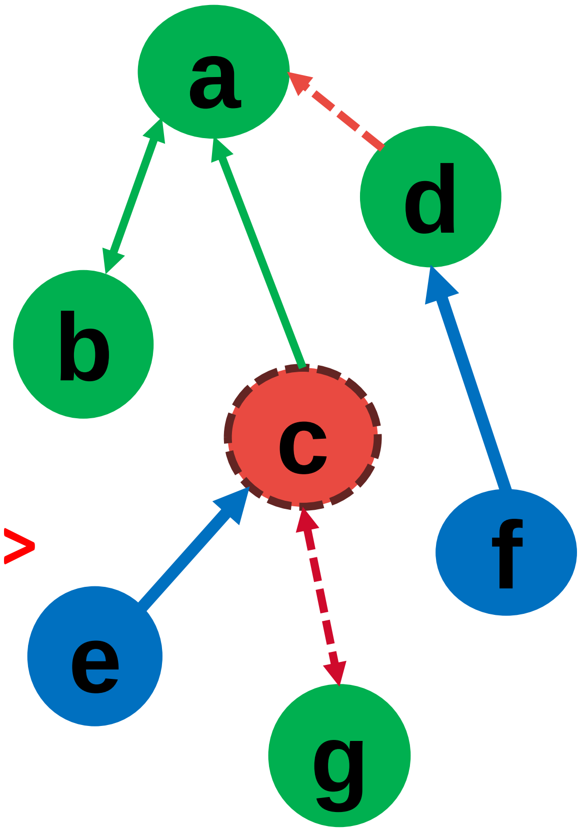
- **Control: possible remedial actions of the agent against possible negative effects of changes**

Control Argumentation Frameworks (CAFs)

- Implementation of self-adaptive systems ensuring real time control tasks in different contexts such as :
  - autonomous driving
  - smart homes
  - surveillance of buildings and streets
  - personalized self-regulation services for humans
  - recommendation policies in finance
  - risk management
  - etc.

Control Argumentation Frameworks (CAFs)

CAF Theory



F: < A_F={a,b,d,g}, $\longrightarrow$ ={(a,b),(b,a),(c,a)}>

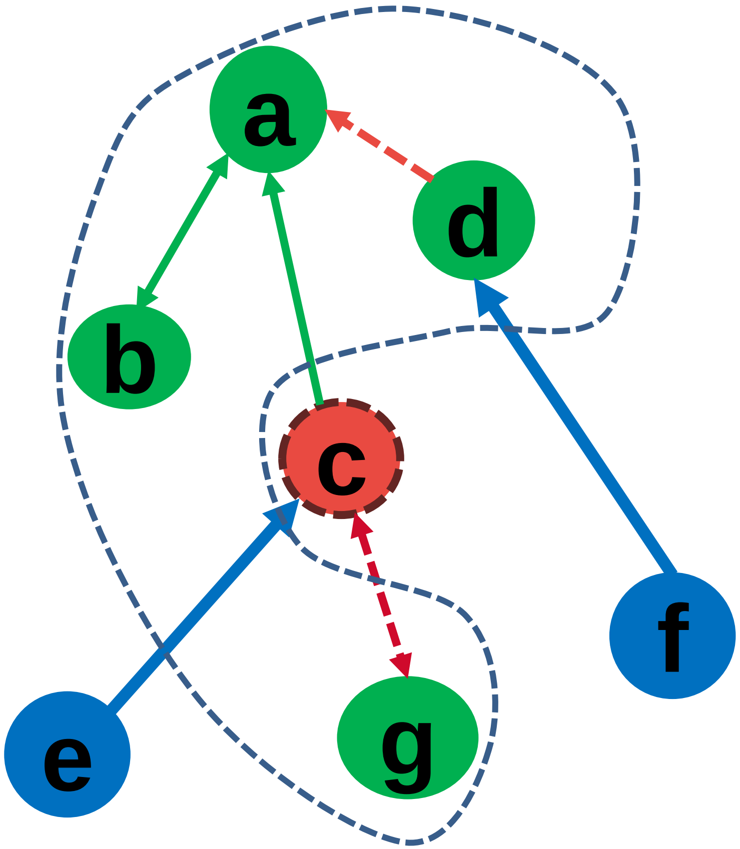
U: < A_U={c}, $\rightleftarrows$ ={(c,g),(g,c)}, $\dashrightarrow$ ={(d,a)}>

C: < A_C={e,f}, $\rightarrow$ ={(e,c},(f,d})>

Control Argumentation Frameworks (CAFs)

Dynamics Under Uncertainty + Computational Methods

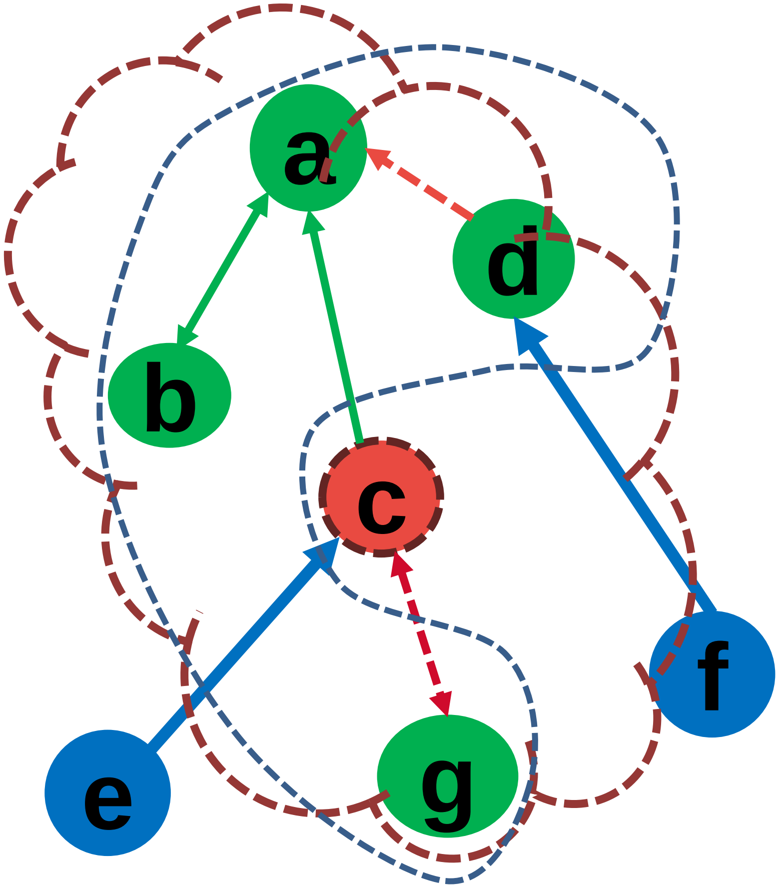
Completions



Control Argumentation Frameworks (CAFs)

Dynamics Under Uncertainty + Computational Methods

Completions



Control Argumentation Frameworks (CAFs)

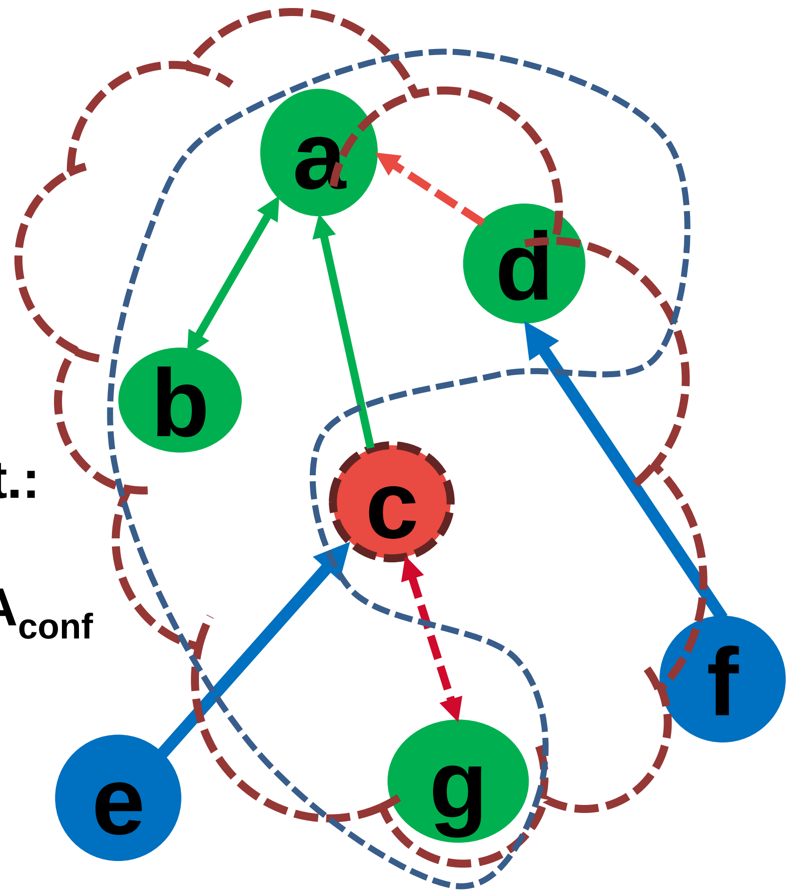
Dynamics Under Uncertainty + Computational Methods

Completions

Controllable $\mathcal{CAF}$s

Given a target $T \subseteq A_F$ and a semantics σ $\mathcal{CAF}$ is skeptically (resp. credulously) controllable w.r.t. T and σ if $\exists A_{conf} \subseteq A_C$ s.t.:

- $\mathcal{CAF}'$ is the result of configuring $\mathcal{CAF}$ by A_{conf}
- T is included in every (resp. at least one) σ-extension of every completion of $\mathcal{CAF}'$



T={a}

Control Argumentation Frameworks (CAFs)

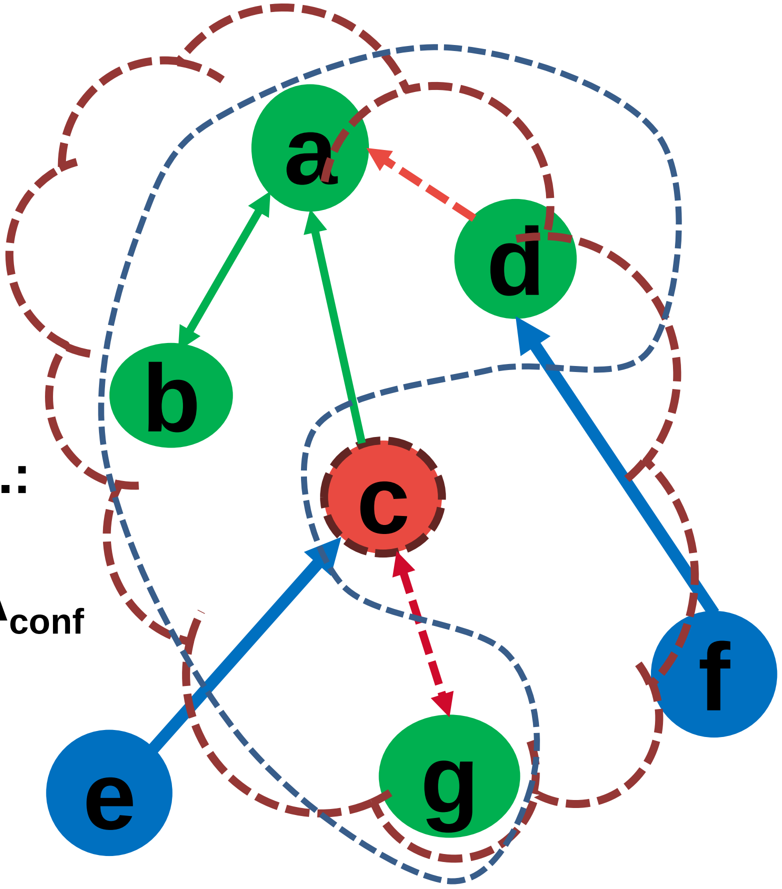
Dynamics Under Uncertainty + Computational Methods

Completions

Controllable *CAF*s

Given a target T $\subseteq$ A_F and a semantics σ *CAF* is skeptically (resp. credulously) controllable w.r.t. T and σ if $\exists A_{conf} \subseteq A_C$ s.t.:

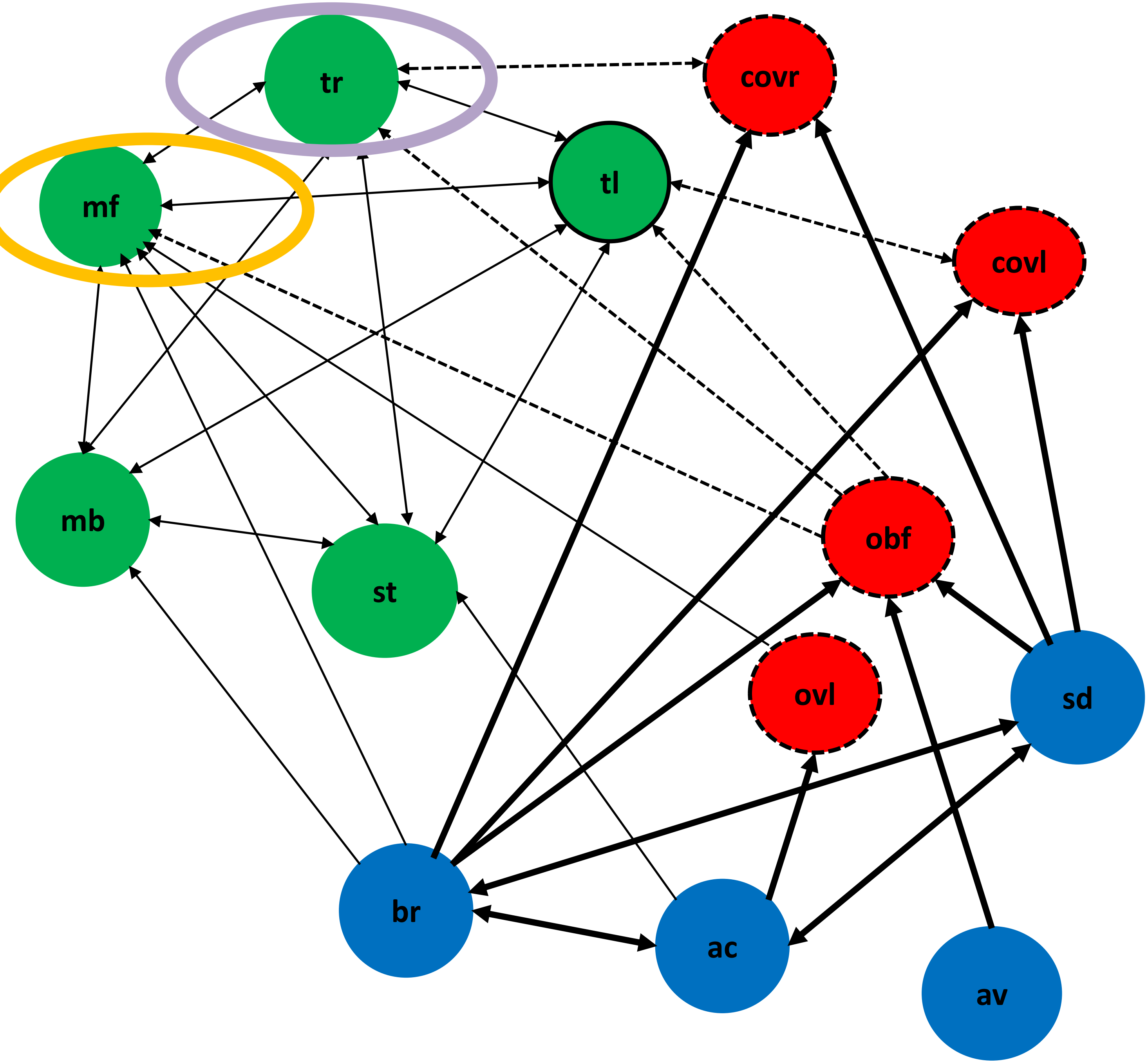
- *CAF'* is the result of configuring *CAF* by A_{conf}
- T is included in every (resp. at least one) σ-extension of every completion of *CAF'*



Use of a QBF-based method

T={a} $\mathcal{E}$={a,e,f,g}

CAFs Application to Autonomous Driving



mf: move forward
mb: move backward
tr: turn right
tl: turn left
st: stop
covl: car overatakes left
covr: car overtakes right
ovl: overtake left
obf: obstacle in front
br: brake
ac: accelerate
av: avoid
sd: slow down

T={mf}

```
<terminated> Main [Java Applic
c lhbr....................:
c subsumption.......... :
c preprocessor......... :
c variable elimination.. :
c upla................. :
c solve................ :
c maxmode.............. :
c solvemode............ :
c cpu time (loading CNF).:
c =======================

c #literals...........: 0
c #clauses............: 0
c #decisions..........: 0
c #BCP operations.....: 16
c #conflicts..........: 0
c #solutions..........: 1
c #pureLiterals.......: 0
c #dcLiterals.........: 0
c #restarts...........: 0
c #restartAttempts....: 0
c #clsimplifications..: 0
c #cusimplifications..: 0
c #lhbr clauses.......: 0
c #inprocessings......: 0
c cpu time............: 0.002s
s SATISFIABLE
Control Configuration: c_arg(avoid). c_arg(accelerate). c_arg(slowDown).
Duration = 193ms
```

T={tr}

```
c preprocessor......... : 1
c variable elimination.. : 1
c upla................. : 1
c solve................ : 1
c maxmode.............. : 1
c solvemode............ : QBF
c cpu time (loading CNF).: 0.002s
c ==========================================================

c #literals...........: 0
c #clauses............: 0
c #decisions..........: 0
c #BCP operations.....: 163
c #conflicts..........: 0
c #solutions..........: 1
c #pureLiterals.......: 0
c #dcLiterals.........: 0
c #restarts...........: 0
c #restartAttempts....: 0
c #clsimplifications..: 0
c #cusimplifications..: 0
c #lhbr clauses.......: 0
c #inprocessings......: 0
c cpu time............: 0.003s
s SATISFIABLE
Control Configuration: c_arg(avoid). c_arg(brake). c_arg(slowDown)
Duration = 270ms
```

Structured Argumentation Frameworks

Logic Programming with Priorities (LPP)

[Kakas&Moraitis, (AAMAS03)]

- Logic Programming without Negation as Failure (LPwNF) (Kakas, Mancarella, Dung, ICLP94; Dimopoulos&Kakas, ILPS95)
  - In the LPwNF logic programs are non-monotonic theories
  - Each logic program is viewed as pool of default sentences from which we must select a suitable subset, called extension to reason with
  - Sentences in a logic program are written in the usual logic programming language with the addition of an explicit negation but without the NAF operator (i.e. not)

Logic Programming with Priorities

- A theory is a pair (T,P) whose sentences are formulae, in a background monotonic logic (L, ⊢), of the form **$L \leftarrow L_1,\ldots,L_n$,** where L, $L_1,\ldots,L_n$ are positive or negative ground literals

- For rules in P the head L refers to an (irreflexive) ***higher-priority*** relation. L has the general form ***$L=h\text{-}p(rule_1, rule_2)$*** where $rule_1$ and $rule_2$ are unique names of rules in the theory

- The derivability relation, ⊢, of the background logic is given by the single inference rule of modus ponens

Logic Programming with Priorities

An LPP theory T is a tuple $\mathbf{T=(\mathcal{T},\mathcal{P})}$ where :

- $\mathcal{T}$ is a set of object level arguments supporting a set of options O
- $\mathcal{P}$ is a set of priority arguments that is partitioned into a finite set of levels, $\mathcal{P}=(\mathcal{P}_1,\ldots\mathcal{P}_n)$
- All the arguments in $\mathcal{P}_1$ are priority arguments $\mathbf{p^1_{12}(arg_1 \succ arg_2)}$, supporting preferences between arguments $\mathbf{arg_1, arg_2} \in \mathcal{T}$
- For any $1 < k \leq n$, all arguments in $\mathcal{P}_k$ are priority arguments, $\mathbf{p^k_{12}(q_1 \succ q_2)}$, supporting a preference between $\mathbf{q_1,q_2} \in \mathcal{P}_{k-1}$

Logic Programming with Priorities

- **Object level rules**:
  - r_i: $L \leftarrow L_1, \ldots, L_n$
  - r_j: $\neg L \leftarrow L_1, \ldots, L_m$
- **Higher Priority rules:**
  - p^1_{ij}: h-p(r_i, r_j) $\leftarrow$ true (i.e. generally) (or conditions$_{ij}$)
  - p^1_{ji} : h-p(r_j, r_i) $\leftarrow$ conditions$_{ji}$
  - p^2_{ji} : h-p(p^1_{ji}, p^1_{ij}) $\leftarrow$ true (or conditions$_{ji}$)
  - p^2_{ij} : h-p(p^1_{ij},p^1_{ji}) $\leftarrow$ conditions$_{ij}$
  - …….

Gorgias* Argumentation Technology

- **Principled Declarative Problem Solving via Argumentation**
  - **Solid theoretical foundation for building acceptable arguments:**
    - **Argument(s) for one option is (are) strong enough to defend against all its (their) counter-arguments for other options**

Policy Compliance ⇔ Acceptable Arguments

*Greek Sophist c.485 — c.380 BCE

Policy Decisions: Challenges

- **Policy Compliance of systems, robust under incomplete & inconsistent information in:**
  - **Dynamic environments**
    - Internal conflicts inside a policy
  - **Multi-policy environments**
    - External conflicts across policies

- **Flexibility in Development of systems**
  - **Modular Adaptation to changes in policies**
  - **Accommodate new policies**

Policy Decisions: Challenges

- **Explainability of Systems**
  - New **EU law** to give everyone a **right to** an **explanation** of any decision affecting them that has been **reached algorithmically**.
  - **Explain level of access granted.**

The Gorgias System (2004 -…)

- Builds sound preferred acceptable arguments from expert/policy knowledge.
- Realizes Decision Making through argumentation for application problems
- Flexible and Robust system
  - Incomplete, contextual and conflicting knowledge
  - Consideration of different (conflicting) view points
- Scenario-based knowledge engineering
- Real-life applications since 2004

Gorgias Application Approach

- Knowledge as Argument Schemes via Scenarios

- Knowledge acquired by:
  - Elicited from Experts
  - Machine Learned
  - Hybrid Acquisition

- Knowledge types:
  - Expert
  - Common Sense
  - Personal biases

Real-life Applications of Gorgias

- **2004-:** Deep Vein Thromboses medical support, Product Pricing, Assisted Living, Sensor Conflict Resolution, Network Security, Cognitive Assistants, Printed/Handwritten Text Discrimination, Eye-Clinic Support System
- **2017: MEDICA – Regulate Data Access**
  - **DEMO: Automating Legislation** for **access to patient data**
- **2017: Data Share Agreements (for health data)**
  - **Coco Cloud: EU project at Imperial College.**
- **2018: Cyber attack management…**

Medical Data Access/Sharing

- **Problem:** Decide **Level of Access** according to **user** and **current circumstances**

- **There are 6 Access Levels (Read & Write)**
  - **Full Access**
  - **Read Only Access**
  - **Suspended Access**
  - **Partial Access**
  - **Restricted Read Access**
  - **No Access**

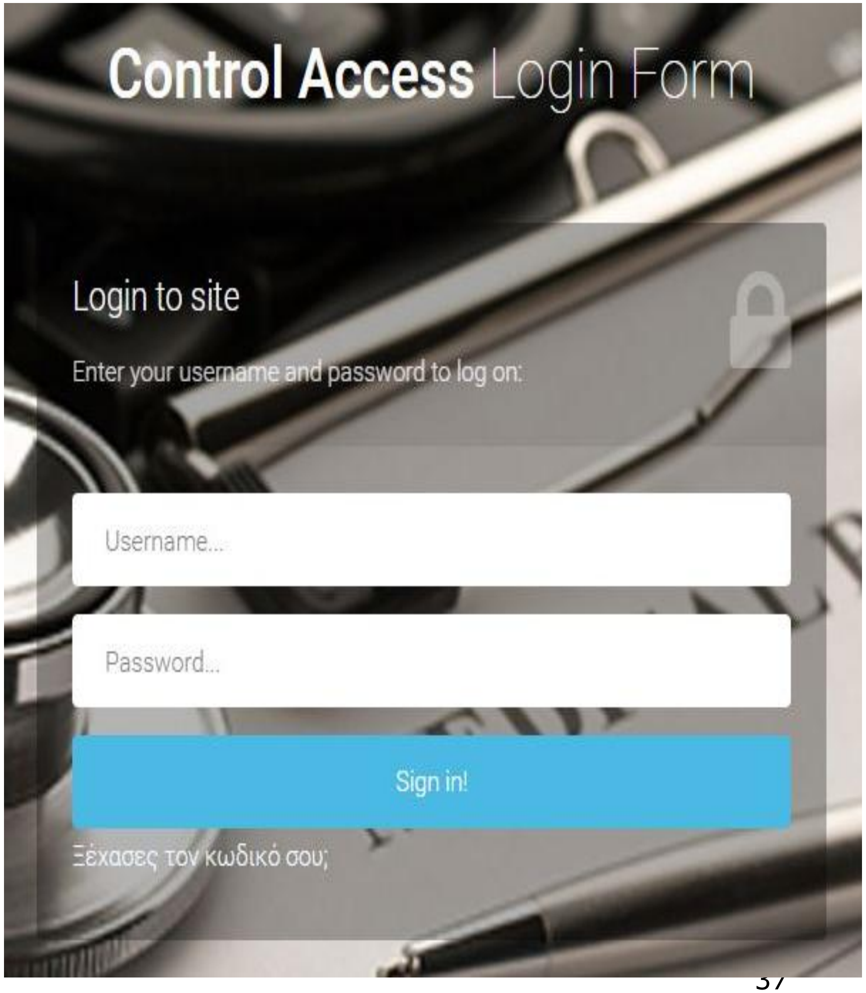
Law 138(I)/2001: Personal Data Protection

Law N. 1(I)/2005: Patient Rights

Medical Data Access: MEDICA

- **MEDICA:**

http://medica.cs.ucy.ac.cy



- **Demo Online**

- **Pilot evaluation**

Medical Data Access

Scenarios \ Options	O_2=deny_access	O_1=allow_access
S_1={sick, privateData}		X
S_2={hospitalDoctor, privateData}	X	
S^1_{21}={hospitalDoctor, privateData, sick}	X	
S^1_{12}={hospitalDoctor, privateData, sick} ∪ {hospitalized}		X
S^2_{21}={hospitalized, hospitalDoctor, privateData, sick } ∪ {unconscious}	X	
S^3_{12}={emergency, hospitalized, hospitalDoctor, privateData, unconscious} ∪ {permision}		X

Decision policy of the agent

r_1: allowAccess$\leftarrow$sick,privateData

r_2: denyAccess$\leftarrow$hospitalDoctor,privateData

p^1_{21} : h-p(r_2,r_1) $\leftarrow$true

p^1_{12}: h-p(r_1,r_2) $\leftarrow$ hospitalized

p^2_{12}: h-p(p^1_{12},p^1_{21}) $\leftarrow$true

p^2_{21}: h-p(p^1_{21},p^1_{12})$\leftarrow$unconscious

p^3_{21}: h-p(p^2_{21} ,p^2_{12}) $\leftarrow$true

p^3_{12}: h-p(p^2_{12},p^2_{21}) $\leftarrow$permision

p^4_{12} : h-p(p^3_{12},p^3_{21}) $\leftarrow$true

Eye Clinic Cognitive Assistant

- **Provides a first level support to patients at the reception of the clinic:**
  - Finds **most expertly probable** diseases
  - Able to recognize the possibility of **severe/urgent** diseases
  - Suggests **extra information/tests** needed to **focus** on the **probable disease**.

Eye Clinic Cognitive Assistant

- **Human-like interaction** with **patients and/or nurse** receptionist:
  - **Input: Symptoms & test** results of patient in their **natural form.**
  - **Output:**
    - **Naturally presented** probable disease(s), urgency level and further tests when needed
    - **Non-technical explanation** of diagnosis

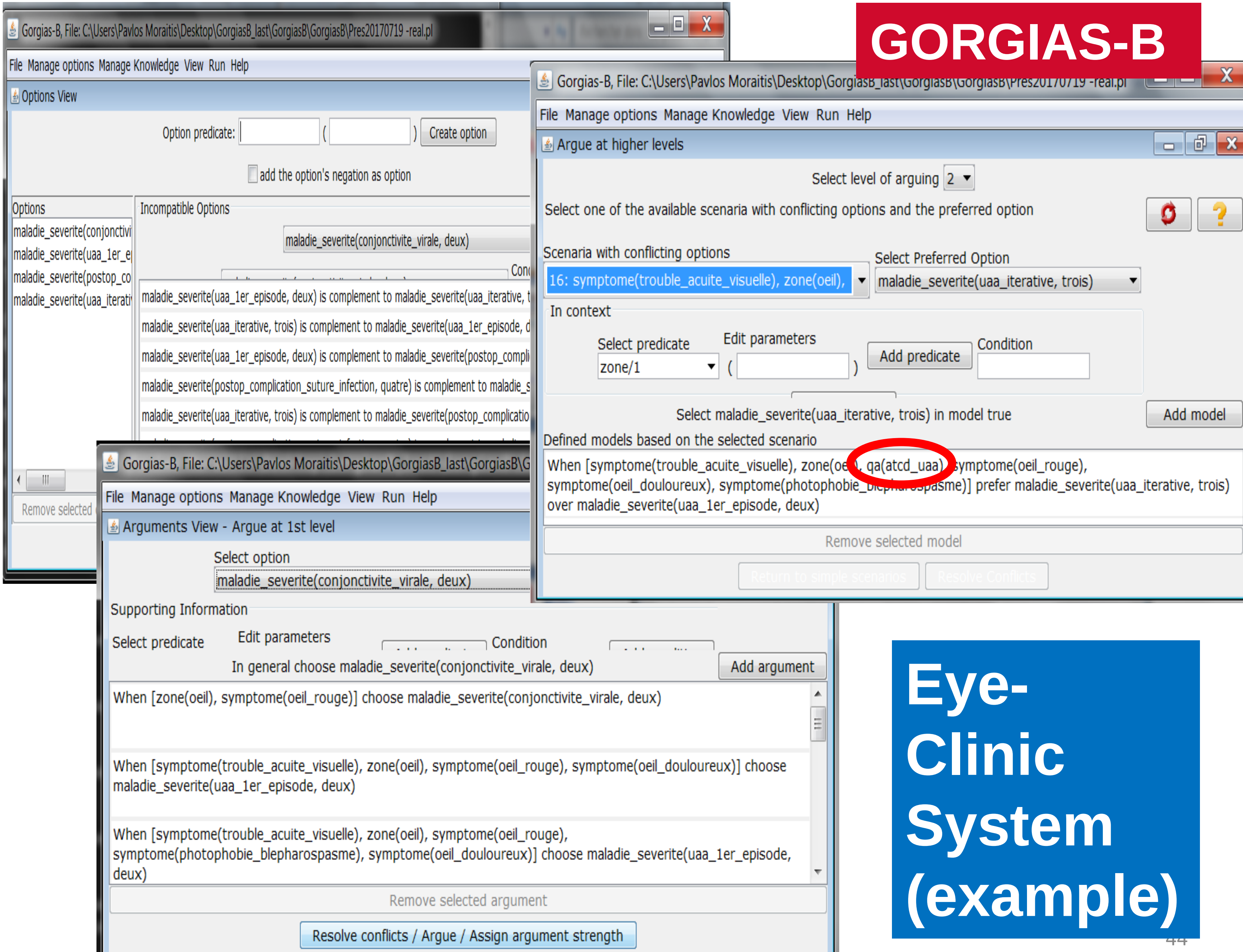
Eye Clinic Cognitive Assistant

- Scale of full expert knowledge:
  - 80 diseases
  - Many Hundreds of scenarios
  - 35+ Parameters (symptoms, zones, contexts)
  - 3-4 Levels of hierarchy of scenarios
  - Several Hypotheticals

Eye-Clinic System Scenarios

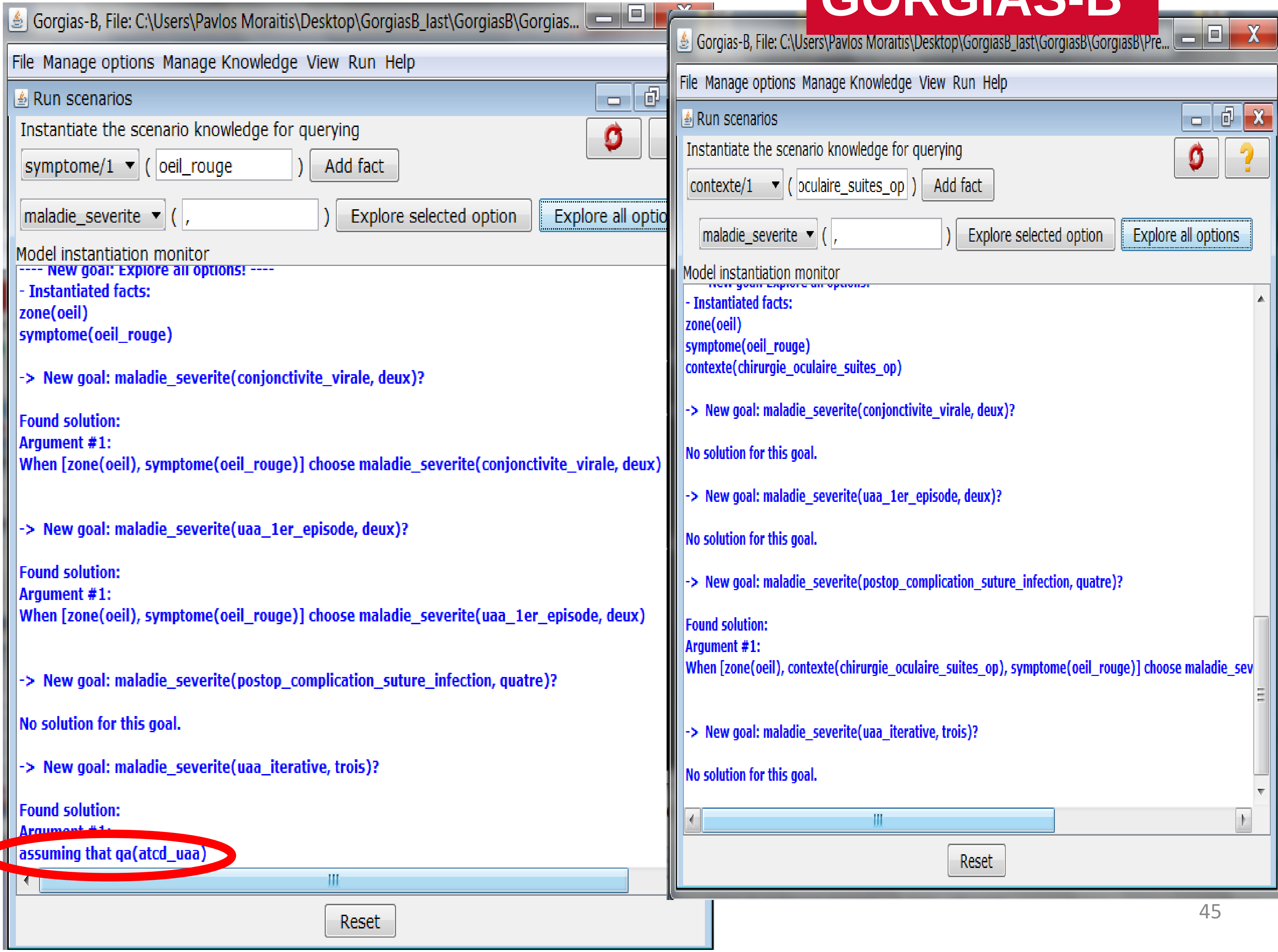
Scenarios \Diseases	D_i	D_j
$S_i=\{s(1),s(2),s(3)\}$	X	
$S_j=\{s(1),s(3),s(4)\}$		X
$S^1_{ij}=\{s(1),s(2),s(3),s(4)\}$	X	
$S^1_{ji}=\{s(1),s(2),s(3),s(4)\}$		X
$S^2_{ji}=\{s(1),s(2),s(3),s(4)\} \cup \{s(8)\}$		X
$S^2_{ij}=\{s(1),s(2),s(3),s(4)\} \cup \{s(19)\}$	X	
$S^3_{ij}=\{s(1),s(2),s(3),s(4),s(8),s(19)\}$	X	

GORGIAS-B



Eye-Clinic System (example)

GORGIAS-B


Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorgias...
File Manage options Manage Knowledge View Run Help
Run scenarios
Instantiate the scenario knowledge for querying
symptome/1 (oeil_rouge) Add fact
maladie_severite (,) Explore selected option Explore all optio
Model instantiation monitor
---- New goal: Explore all options! ----
- Instantiated facts:
zone(oeil)
symptome(oeil_rouge)
-> New goal: maladie_severite(conjonctivite_virale, deux)?
Found solution:
Argument #1:
When [zone(oeil), symptome(oeil_rouge)] choose maladie_severite(conjonctivite_virale, deux)
-> New goal: maladie_severite(uaa_1er_episode, deux)?
Found solution:
Argument #1:
When [zone(oeil), symptome(oeil_rouge)] choose maladie_severite(uaa_1er_episode, deux)
-> New goal: maladie_severite(postop_complication_suture_infection, quatre)?
No solution for this goal.
-> New goal: maladie_severite(uaa_iterative, trois)?
Found solution:
assuming that qa(atcd_uaa)
Reset
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\GorgiasB\Pre...
File Manage options Manage Knowledge View Run Help
Run scenarios
Instantiate the scenario knowledge for querying
contexte/1 (oculaire_suites_op) Add fact
maladie_severite (,) Explore selected option Explore all options
Model instantiation monitor
- Instantiated facts:
zone(oeil)
symptome(oeil_rouge)
contexte(chirurgie_oculaire_suites_op)
-> New goal: maladie_severite(conjonctivite_virale, deux)?
No solution for this goal.
-> New goal: maladie_severite(uaa_1er_episode, deux)?
No solution for this goal.
-> New goal: maladie_severite(postop_complication_suture_infection, quatre)?
Found solution:
Argument #1:
When [zone(oeil), contexte(chirurgie_oculaire_suites_op), symptome(oeil_rouge)] choose maladie_sev
-> New goal: maladie_severite(uaa_iterative, trois)?
No solution for this goal.
Reset

GORGIAS-B

Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\GorgiasB\Pres20170719 -real.pl

File Manage options Manage Knowledge View Run Help

Gorgias file

Pres20170719 -real.pl

```
rule(r1(conjonctivite_virale, deux), maladie_severite(conjonctivite_virale, deux), []):-zone(oeil), symptome(oeil_rouge).
rule(r2(uaa_1er_episode, deux), maladie_severite(uaa_1er_episode, deux), []):-zone(oeil), symptome(trouble_acuite_visuelle), symptome(oeil_douloureux), symptome(oeil_rouge).
rule(r3(uaa_1er_episode, deux), maladie_severite(uaa_1er_episode, deux), []):-symptome(photophobie_blepharospasme), zone(oeil), symptome(trouble_acuite_visuelle), symptome(oeil_
rule(r4(postop_complication_suture_infection, quatre), maladie_severite(postop_complication_suture_infection, quatre), []):-zone(oeil), symptome(trouble_acuite_visuelle), symptome(oeil
rule(r5(postop_complication_suture_infection, quatre), maladie_severite(postop_complication_suture_infection, quatre), []):-zone(oeil), symptome(trouble_acuite_visuelle), symptome(oeil
rule(r6(conjonctivite_virale, deux), maladie_severite(conjonctivite_virale, deux), []):-zone(oeil), symptome(oeil_rouge), symptome(oeil_colle), symptome(oedeme_conjonctival), symptom
rule(r7(uaa_1er_episode, deux), maladie_severite(uaa_1er_episode, deux), []):-zone(oeil), symptome(oeil_rouge).
rule(r8(postop_complication_suture_infection, quatre), maladie_severite(postop_complication_suture_infection, quatre), []):-zone(oeil), symptome(oeil_rouge), contexte(chirurgie_oculaire
rule(r9(postop_complication_suture_infection, quatre), maladie_severite(postop_complication_suture_infection, quatre), []):-zone(oeil), symptome(oeil_rouge), symptome(oeil_douloureux
rule(r10(uaa_iterative, trois), maladie_severite(uaa_iterative, trois), [qa(atcd_uaa)]):-zone(oeil), symptome(oeil_rouge), symptome(oeil_douloureux), symptome(photophobie_blepharosp
rule(r11(uaa_iterative, trois), maladie_severite(uaa_iterative, trois), [qa(atcd_uaa)]):-zone(oeil), symptome(oeil_rouge), symptome(oeil_douloureux), symptome(trouble_acuite_visuelle),
rule(r12(uaa_iterative, trois), maladie_severite(uaa_iterative, trois), [qa(atcd_uaa)]):-zone(oeil), symptome(oeil_rouge).
rule(p1(deux, uaa_1er_episode), prefer(r3(uaa_1er_episode, deux), r1(conjonctivite_virale, deux)), []).
rule(p2(quatre, postop_complication_suture_infection), prefer(r4(postop_complication_suture_infection, quatre), r1(conjonctivite_virale, deux)), []).
rule(p3(quatre, postop_complication_suture_infection), prefer(r5(postop_complication_suture_infection, quatre), r1(conjonctivite_virale, deux)), []).
rule(p4(quatre, postop_complication_suture_infection), prefer(r4(postop_complication_suture_infection, quatre), r2(uaa_1er_episode, deux)), []).
rule(p5(quatre, postop_complication_suture_infection), prefer(r5(postop_complication_suture_infection, quatre), r2(uaa_1er_episode, deux)), []).
rule(p6(quatre, postop_complication_suture_infection), prefer(r4(postop_complication_suture_infection, quatre), r3(uaa_1er_episode, deux)), []).
rule(p7(quatre, postop_complication_suture_infection), prefer(r5(postop_complication_suture_infection, quatre), r3(uaa_1er_episode, deux)), []).
rule(p8(quatre, postop_complication_suture_infection), prefer(r8(postop_complication_suture_infection, quatre), r1(conjonctivite_virale, deux)), []).
rule(p9(quatre, postop_complication_suture_infection), prefer(r8(postop_complication_suture_infection, quatre), r7(uaa_1er_episode, deux)), []).
rule(p10(deux, uaa_1er_episode), prefer(r2(uaa_1er_episode, deux), r1(conjonctivite_virale, deux)), []).
rule(p11(deux, conjonctivite_virale), prefer(r6(conjonctivite_virale, deux), r7(uaa_1er_episode, deux)), []).
rule(p12(trois, uaa_iterative), prefer(r10(uaa_iterative, trois), r1(conjonctivite_virale, deux)), []).
rule(p13(trois, uaa_iterative), prefer(r11(uaa_iterative, trois), r1(conjonctivite_virale, deux)), []).
rule(p14(deux, conjonctivite_virale), prefer(r6(conjonctivite_virale, deux), r2(uaa_1er_episode, deux)), []).
rule(p15(trois, uaa_iterative), prefer(r10(uaa_iterative, trois), r2(uaa_1er_episode, deux)), []).
rule(p16(trois, uaa_iterative), prefer(r11(uaa_iterative, trois), r2(uaa_1er_episode, deux)), []).
rule(p17(deux, conjonctivite_virale), prefer(r6(conjonctivite_virale, deux), r3(uaa_1er_episode, deux)), []).
rule(p18(quatre, postop_complication_suture_infection), prefer(r8(postop_complication_suture_infection, quatre), r3(uaa_1er_episode, deux)), []).
rule(p19(quatre, postop_complication_suture_infection), prefer(r9(postop_complication_suture_infection, quatre), r3(uaa_1er_episode, deux)), []).
rule(p20(trois, uaa_iterative), prefer(r10(uaa_iterative, trois), r3(uaa_1er_episode, deux)), []).
rule(p21(trois, uaa_iterative), prefer(r11(uaa_iterative, trois), r3(uaa_1er_episode, deux)), []).
rule(p22(deux, conjonctivite_virale), prefer(r6(conjonctivite_virale, deux), r4(postop_complication_suture_infection, quatre)), []).
rule(p23(quatre, postop_complication_suture_infection), prefer(r4(postop_complication_suture_infection, quatre), r7(uaa_1er_episode, deux)), []).
rule(p24(quatre, postop_complication_suture_infection), prefer(r4(postop_complication_suture_infection, quatre), r10(uaa_iterative, trois)), []).
```

Automatically Generated Source Code

Call Assistant

- **Requirements**

  Paul wants to train his personal assistant to manage his calls. He wants him to do one out of two options. The first is to allow the phone to ring when there is call, the second is to deny the call. In general he chooses the first option over the second. If he is at work, however, that is a reason to deny the call. When he is at work he prefers to allow family calls over denying them, except when he is in a meeting, when he prefers to deny over allowing the call. Being in a meeting there is a possibility to prefer to accept a call from his son when he is at school. He will accept it if he believes that his son is ill.

Call Assistant Scenarios

Scenarios \ Options	O_1=allow(Call)	O_2=deny(Call)
S^0={phone_call}	X	X
S^1_{12}={phone_call}	X	
S^1_{21}={phone_call} $\cup$ {at_work}		X
S^2_{12}={phone_call, at_work} $\cup$ {familly_member(Call)}	X	
S^3_{21}={phone_call, at_work, familly_member(Call)} $\cup$ {in_meeting}		X
S^3_{12}={phone_call, at_work, familly_member(Call), in_meeting} $\cup${from_son(Call), son_at_school, son_is_ill}	X	

Decision policy of the call assistant agent

r_1: allow(Call)$\leftarrow$phone_call

r_2: deny(Call)$\leftarrow$phone_call

p^1_{12} : h-p(r_1,r_2) $\leftarrow$true

p^1_{21}: h-p(r_2,r_1) $\leftarrow$ at_work

p^2_{21}: h-p(p^1_{21}, p^1_{12}) $\leftarrow$true

p^2_{12}: h-p(p^1_{12},p^1_{21}) $\leftarrow$familly_member(Call)

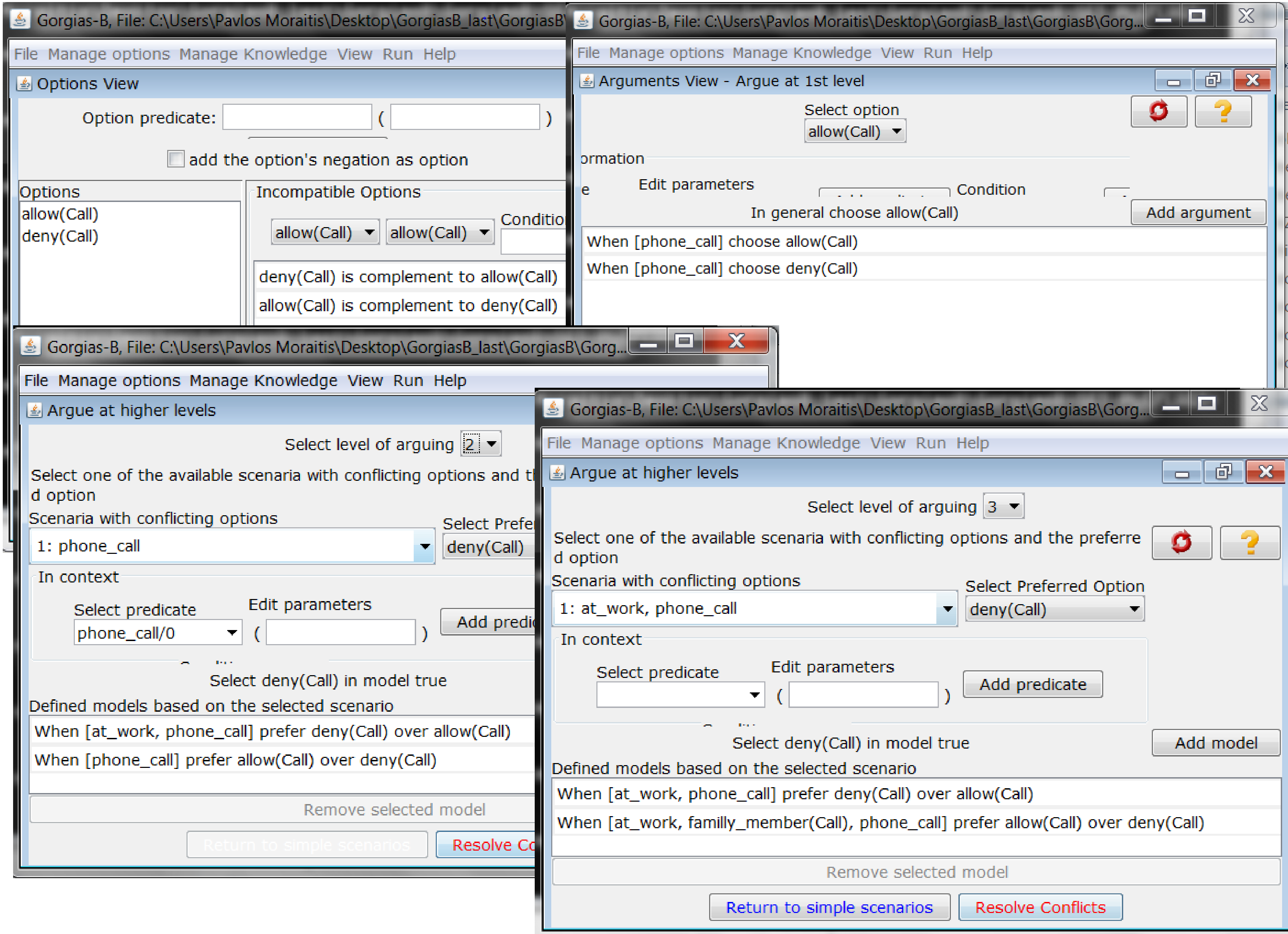
p^3_{12}: h-p(p^2_{12}, p^2_{21}) $\leftarrow$true

p^3_{21}: h-p(p^2_{21}, p^2_{12}) $\leftarrow$at_meeting

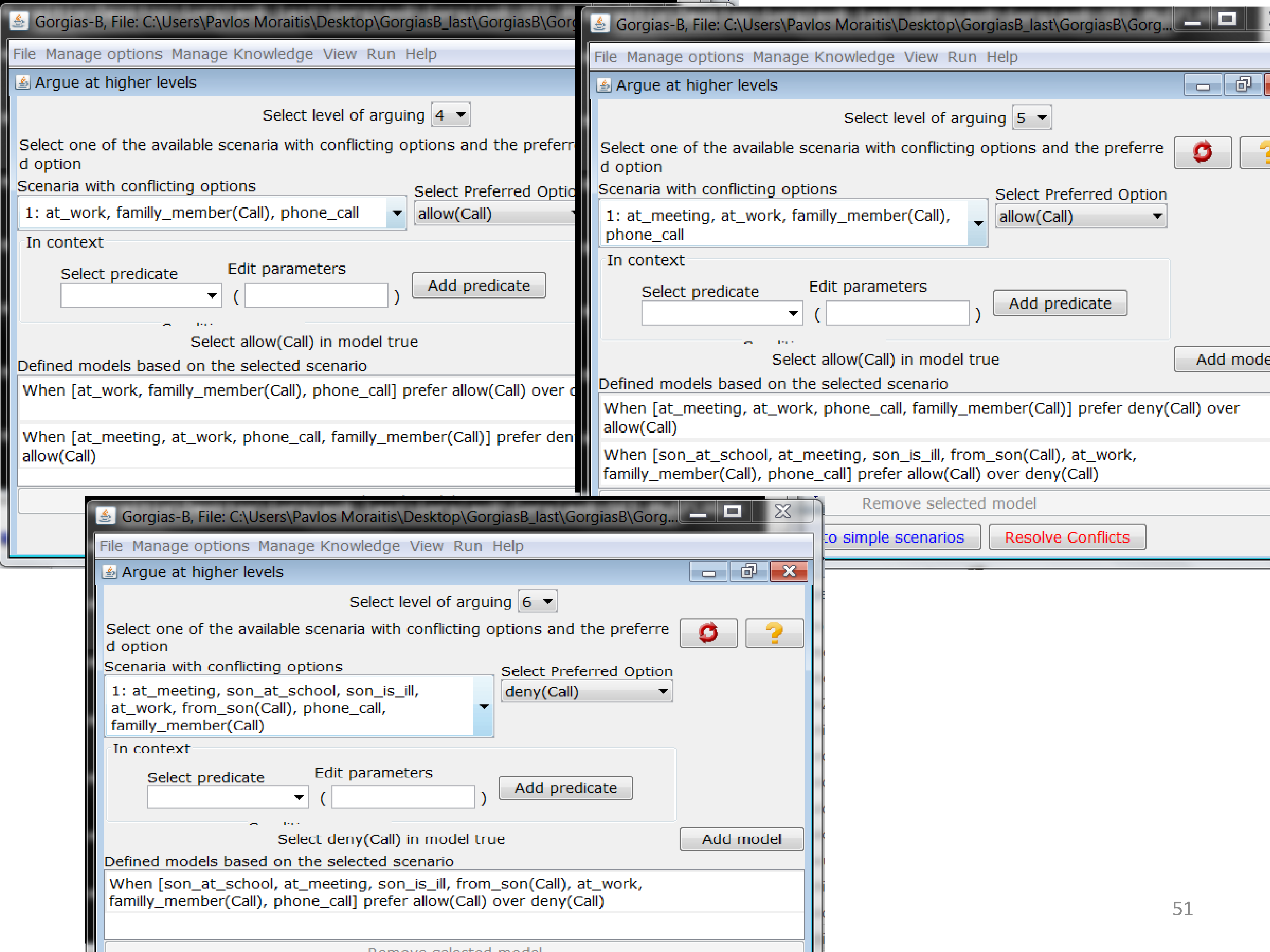
p^4_{21} : h-p(p^3_{21}, p^3_{12}) $\leftarrow$true

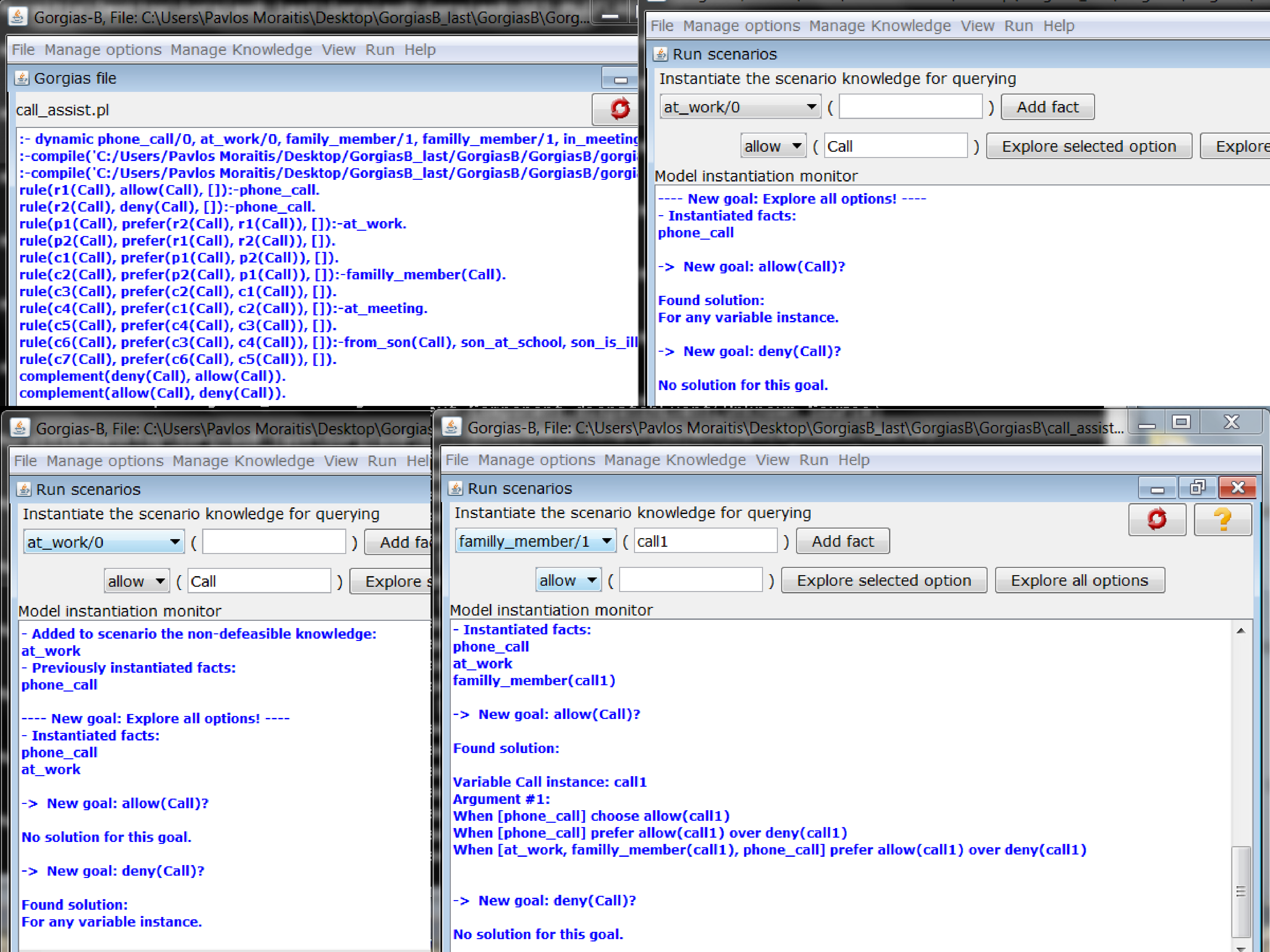
p^4_{12}: h-p(p^3_{12}, p^3_{21}) $\leftarrow$from_son(Call), son_at_school, son_is_ill

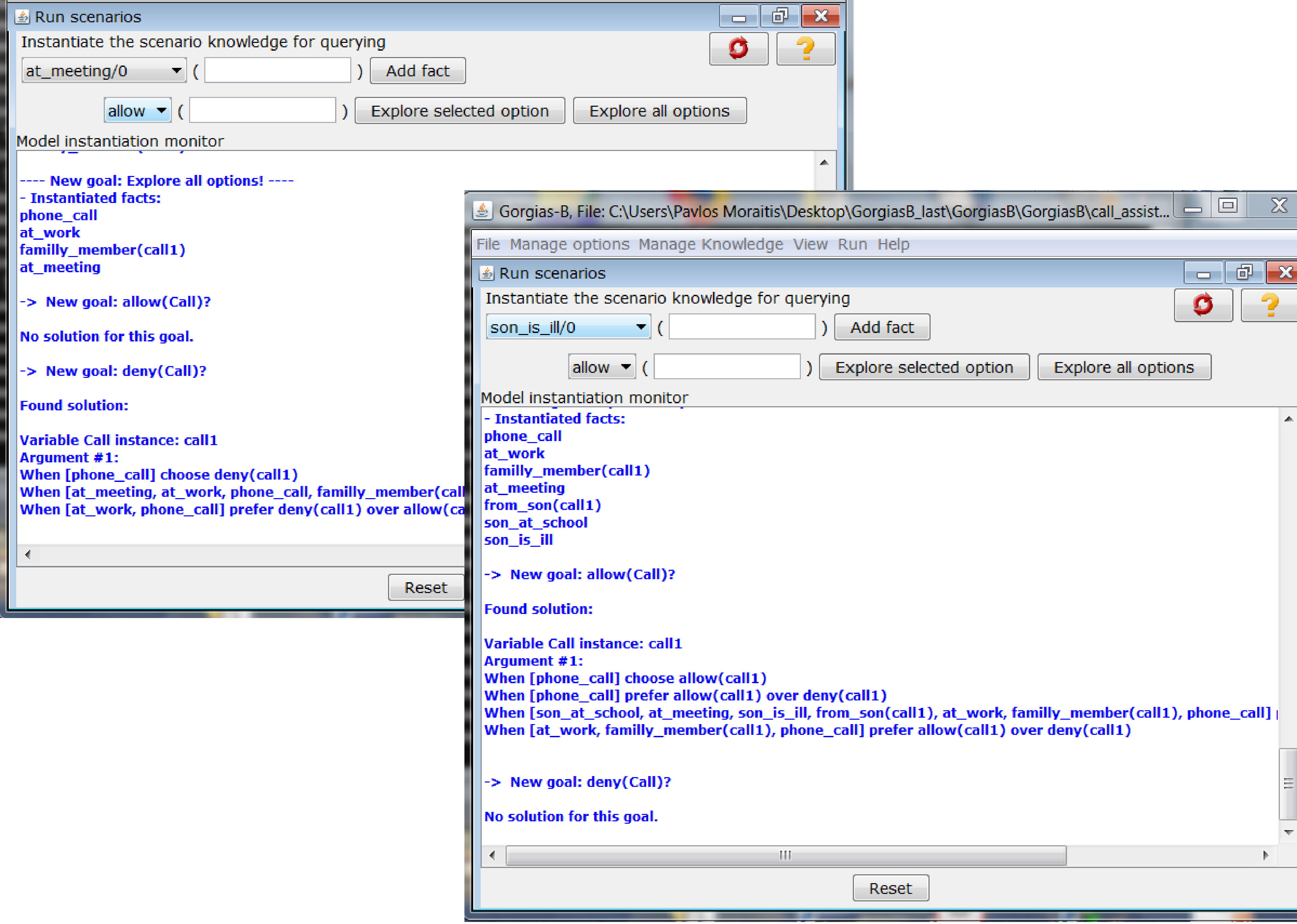
p^5_{12}: h-p(p^4_{12}, p^4_{21}) $\leftarrow$true



Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB
File Manage options Manage Knowledge View Run Help
Options View
Option predicate: ( )
add the option's negation as option
Options
allow(Call)
deny(Call)
Incompatible Options
allow(Call)
allow(Call)
Conditio
deny(Call) is complement to allow(Call)
allow(Call) is complement to deny(Call)
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Arguments View - Argue at 1st level
Select option
allow(Call)
ormation
Edit parameters
Condition
In general choose allow(Call)
Add argument
When [phone_call] choose allow(Call)
When [phone_call] choose deny(Call)
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Argue at higher levels
Select level of arguing 2
Select one of the available scenaria with conflicting options and t
d option
Scenaria with conflicting options
Select Prefe
1: phone_call
deny(Call)
In context
Select predicate
Edit parameters
phone_call/0
Add predi
Select deny(Call) in model true
Defined models based on the selected scenario
When [at_work, phone_call] prefer deny(Call) over allow(Call)
When [phone_call] prefer allow(Call) over deny(Call)
Remove selected model
Return to simple scenarios
Resolve Co
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Argue at higher levels
Select level of arguing 3
Select one of the available scenaria with conflicting options and the preferre
d option
Scenaria with conflicting options
Select Preferred Option
1: at_work, phone_call
deny(Call)
In context
Select predicate
Edit parameters
Add predicate
Select deny(Call) in model true
Add model
Defined models based on the selected scenario
When [at_work, phone_call] prefer deny(Call) over allow(Call)
When [at_work, familly_member(Call), phone_call] prefer allow(Call) over deny(Call)
Remove selected model
Return to simple scenarios
Resolve Conflicts


Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gor
File Manage options Manage Knowledge View Run Help
Argue at higher levels
Select level of arguing 4
Select one of the available scenaria with conflicting options and the preferr
d option
Scenaria with conflicting options
1: at_work, familly_member(Call), phone_call
Select Preferred Optio
allow(Call)
In context
Select predicate
Edit parameters
Add predicate
Select allow(Call) in model true
Defined models based on the selected scenario
When [at_work, familly_member(Call), phone_call] prefer allow(Call) over
When [at_meeting, at_work, phone_call, familly_member(Call)] prefer den
allow(Call)
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Argue at higher levels
Select level of arguing 5
Select one of the available scenaria with conflicting options and the preferre
d option
Scenaria with conflicting options
1: at_meeting, at_work, familly_member(Call),
phone_call
Select Preferred Option
allow(Call)
In context
Select predicate
Edit parameters
Add predicate
Select allow(Call) in model true
Add mode
Defined models based on the selected scenario
When [at_meeting, at_work, phone_call, familly_member(Call)] prefer deny(Call) over
allow(Call)
When [son_at_school, at_meeting, son_is_ill, from_son(Call), at_work,
familly_member(Call), phone_call] prefer allow(Call) over deny(Call)
Remove selected model
to simple scenarios
Resolve Conflicts
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Argue at higher levels
Select level of arguing 6
Select one of the available scenaria with conflicting options and the preferre
d option
Scenaria with conflicting options
1: at_meeting, son_at_school, son_is_ill,
at_work, from_son(Call), phone_call,
familly_member(Call)
Select Preferred Option
deny(Call)
In context
Select predicate
Edit parameters
Add predicate
Select deny(Call) in model true
Add model
Defined models based on the selected scenario
When [son_at_school, at_meeting, son_is_ill, from_son(Call), at_work,
familly_member(Call), phone_call] prefer allow(Call) over deny(Call)


Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\Gorg...
File Manage options Manage Knowledge View Run Help
Gorgias file
call_assist.pl
:- dynamic phone_call/0, at_work/0, family_member/1, familly_member/1, in_meeting
:-compile('C:/Users/Pavlos Moraitis/Desktop/GorgiasB_last/GorgiasB/GorgiasB/gorgi
:-compile('C:/Users/Pavlos Moraitis/Desktop/GorgiasB_last/GorgiasB/GorgiasB/gorgi
rule(r1(Call), allow(Call), []):-phone_call.
rule(r2(Call), deny(Call), []):-phone_call.
rule(p1(Call), prefer(r2(Call), r1(Call)), []):-at_work.
rule(p2(Call), prefer(r1(Call), r2(Call)), []).
rule(c1(Call), prefer(p1(Call), p2(Call)), []).
rule(c2(Call), prefer(p2(Call), p1(Call)), []):-familly_member(Call).
rule(c3(Call), prefer(c2(Call), c1(Call)), []).
rule(c4(Call), prefer(c1(Call), c2(Call)), []):-at_meeting.
rule(c5(Call), prefer(c4(Call), c3(Call)), []).
rule(c6(Call), prefer(c3(Call), c4(Call)), []):-from_son(Call), son_at_school, son_is_ill
rule(c7(Call), prefer(c6(Call), c5(Call)), []).
complement(deny(Call), allow(Call)).
complement(allow(Call), deny(Call)).
File Manage options Manage Knowledge View Run Help
Run scenarios
Instantiate the scenario knowledge for querying
at_work/0 ( ) Add fact
allow (Call) Explore selected option Explore
Model instantiation monitor
---- New goal: Explore all options! ----
- Instantiated facts:
phone_call
-> New goal: allow(Call)?
Found solution:
For any variable instance.
-> New goal: deny(Call)?
No solution for this goal.
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\Gorgias
File Manage options Manage Knowledge View Run Hel
Run scenarios
Instantiate the scenario knowledge for querying
at_work/0 ( ) Add fa
allow (Call) Explore s
Model instantiation monitor
- Added to scenario the non-defeasible knowledge:
at_work
- Previously instantiated facts:
phone_call
---- New goal: Explore all options! ----
- Instantiated facts:
phone_call
at_work
-> New goal: allow(Call)?
No solution for this goal.
-> New goal: deny(Call)?
Found solution:
For any variable instance.
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\GorgiasB\call_assist...
File Manage options Manage Knowledge View Run Help
Run scenarios
Instantiate the scenario knowledge for querying
familly_member/1 (call1) Add fact
allow ( ) Explore selected option Explore all options
Model instantiation monitor
- Instantiated facts:
phone_call
at_work
familly_member(call1)
-> New goal: allow(Call)?
Found solution:
Variable Call instance: call1
Argument #1:
When [phone_call] choose allow(call1)
When [phone_call] prefer allow(call1) over deny(call1)
When [at_work, familly_member(call1), phone_call] prefer allow(call1) over deny(call1)
-> New goal: deny(Call)?
No solution for this goal.

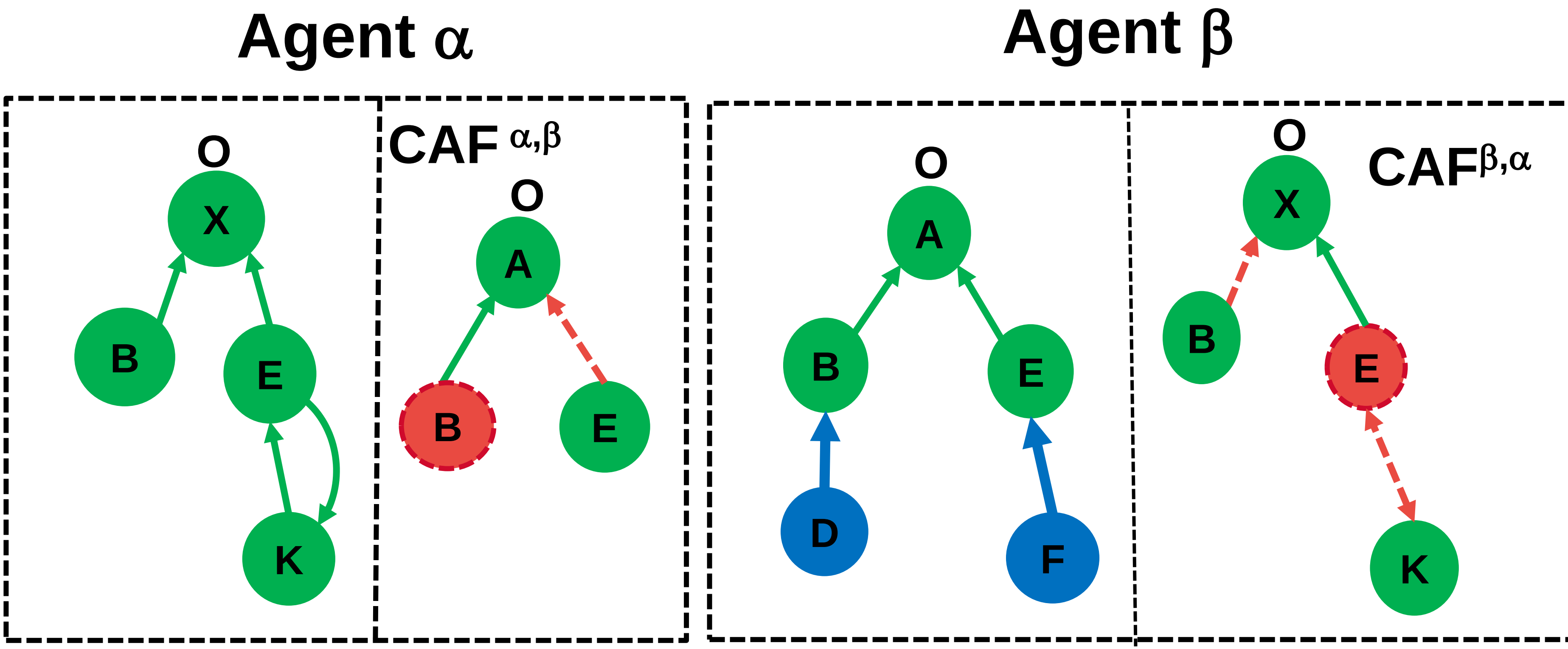

Run scenarios
Instantiate the scenario knowledge for querying
at_meeting/0
Add fact
allow
Explore selected option
Explore all options
Model instantiation monitor
---- New goal: Explore all options! ----
- Instantiated facts:
phone_call
at_work
familly_member(call1)
at_meeting
-> New goal: allow(Call)?
No solution for this goal.
-> New goal: deny(Call)?
Found solution:
Variable Call instance: call1
Argument #1:
When [phone_call] choose deny(call1)
When [at_meeting, at_work, phone_call, familly_member(call
When [at_work, phone_call] prefer deny(call1) over allow(ca
Reset
Gorgias-B, File: C:\Users\Pavlos Moraitis\Desktop\GorgiasB_last\GorgiasB\GorgiasB\call_assist...
File Manage options Manage Knowledge View Run Help
Run scenarios
Instantiate the scenario knowledge for querying
son_is_ill/0
Add fact
allow
Explore selected option
Explore all options
Model instantiation monitor
- Instantiated facts:
phone_call
at_work
familly_member(call1)
at_meeting
from_son(call1)
son_at_school
son_is_ill
-> New goal: allow(Call)?
Found solution:
Variable Call instance: call1
Argument #1:
When [phone_call] choose allow(call1)
When [phone_call] prefer allow(call1) over deny(call1)
When [son_at_school, at_meeting, son_is_ill, from_son(call1), at_work, familly_member(call1), phone_call]
When [at_work, familly_member(call1), phone_call] prefer allow(call1) over deny(call1)
-> New goal: deny(Call)?
No solution for this goal.
Reset

Argumentation-based Automated Negotiation

- Conflict resolution concerning a specific issue related to a resource sharing (e.g. the price of a product)
- Agents exchange offers supported by arguments
- Search for an agreement through the exchanged arguments
- Proponents (agents) defend the supporting arguments by attacking the opponents arguments that attack them, etc.

Negotiation with CAFs

[Dimopoulos, Mailly, Moraitis (AAMAS19)]

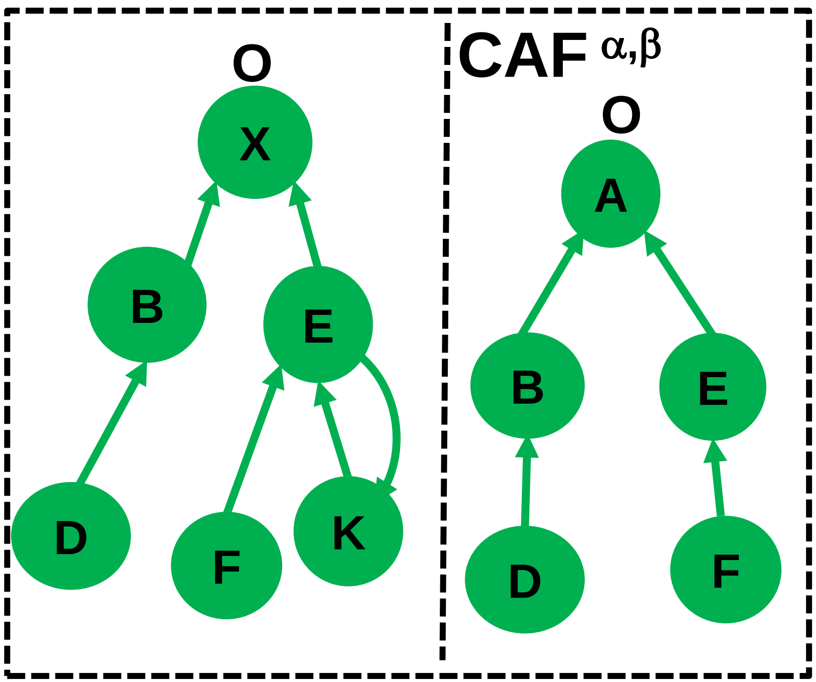


Initial theories of agents α and β: each agent uses a CAF for representing the incomplete knowledge he has about the profile of his opponent

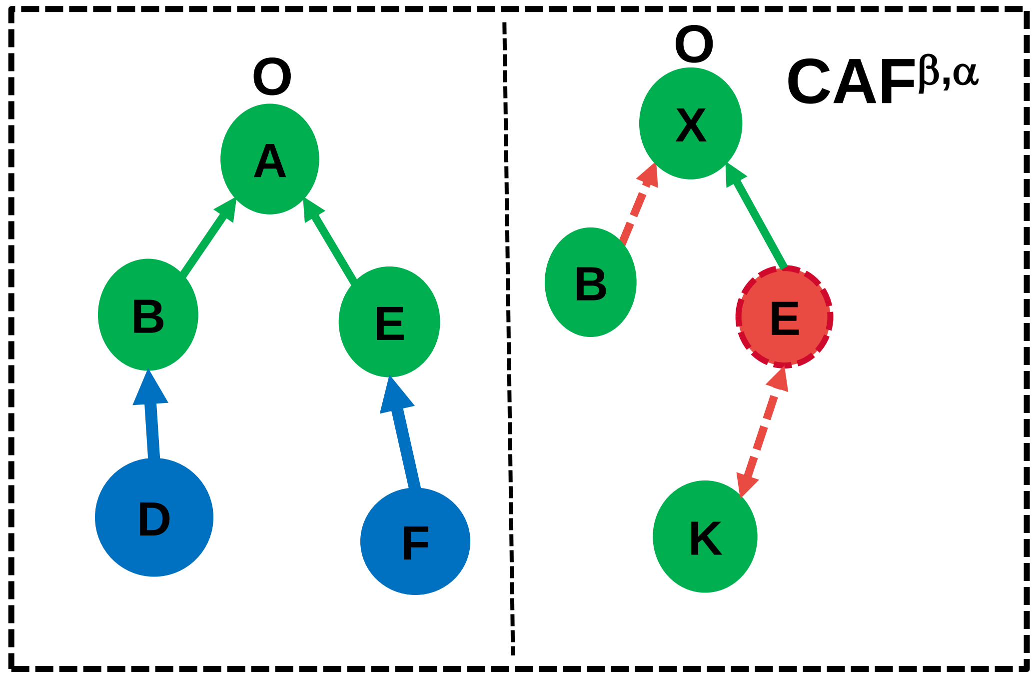
Bidding Strategy

Theories after proposal of offer O (supported by argument X and control arguments D and F) by agent β

Agent α



Agent β



The goal of agent β is to persuade agent α to accept the supporting argument X (and therefore offer O) by defending argument X with the control arguments D and F

The Negotiation Dialogue between Agent α and Agent β

Conclusions

- **Computational argumentation is now mature enough for real world applications**
- **CAFs are very well suited for modeling self-adaptive systems**
- **LPP and GORGIAS are very well suited for modeling decision policies under incomplete, contextual and conflicting knowledge**

References

- **Related to the application of LPP-GORGIAS and CAFs in single agent reasoning**

- Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence. 77(2):321–358.
- Dimopoulos Y., Mailly JG., Moraitis P., "Control Argumentation Frameworks", in 32nd AAAI Conference on Artificial Intelligence (AAAI'18), pp. 4678-4685, New Orleans, USA, 2018
- A.C. Kakas, P. Mancarella, and P.M. Dung, "The acceptability semantics for logic programs", in Proceedings of ICLP94, pp. 504-519, 1994
- Dimopoulos, Y., Kakas, A., "Logic Programming without Negation as Failure", in ILPS95: 369-383, 1995
- Kakas, A.C., Moraitis, P., "Argumentation based decision making for autonomous agents", in 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03), pp. 883-890, Melbourne, Australia, 2003
- Kakas A., Moraitis P., Spanoudakis N., "GORGIAS: Applying argumentation", Argument & Computation, Vol. 10, No. 1, pp. 55-81, 2019

References (cont.)

- **Related to the application of LPP-GORGIAS and CAFs in multi-agent systems (i.e. agent dialogues)**

- **Dimopoulos Y., Mailly JG., Moraitis P., "Argumentation-based Negotiation with Incomplete Opponent Profiles", in 18th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'19), Montreal , Canada, 2019.**
- **Kakas A., Moraitis P., "Adaptive Agent Negotiation via Argumentation", in 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06), pp. 384-391, Hakodate, Japan, 2006**
- **Kakas, A., Maudet, N., Moraitis, P. "Modular Representation of Agent Interaction Rules through Argumentation", in Journal of Autonomous Agents and Multi-Agent Systems, (JAAMAS), Springer, vol. 11, no. 2, pp. 189-206, 2005**