A PROLOG META-INTERPRETER FOR AGENTSPEAK(L)

M.Sc. THESIS

of

Dimitrios Dimitriadis

Thessaloniki, 21 June 2019

PRESENTATION OUTLINE

- Introduction
- Syntax of AgentSpeak(L)
- Implemented Features
- Implementation
  - The Parser
  - The Solver
  - Design of the Meta-Interpreter
  - Failure Handling Mechanism
- Agents' Communication
- Test Cases
- Conclusions & Future Work

INTRODUCTION

- Software Agents Attributes
  - Autonomy
  - Reactivity
  - Social ability
- BDI Agents (Belief – Desire – Intention)
- AgentSpeak(L)
  - Jason (An AgentSpeak(L) JAVA based Meta-Interpreter)

SYNTAX OF AGENTSPEAK(L)

SYNTAX OF AGENTSPEAK(L)

- Beliefs
  - `light(green)`
  - `temperature(32)`

- Mental Rules
  -
```
findMaxTemp(Temp):-
        temp(Temp) &
        not ( (temp(Y) & Y > Temp) ).
```

SYNTAX OF AGENTSPEAK(L)

- Goals
  - Achievement goals

    ```
    !close_valve
    !shut_window
    ```

  - Test goals

    ```
    ?has_leak
    ?door_open
    ```

SYNTAX OF AGENTSPEAK(L)

- Events
  - Belief < Addition / Deletion

```
-light(red)
+light(green)
```

  - Achievement / Test > Goal Addition

```
+!close_valve
+?has_leak
```

SYNTAX OF AGENTSPEAK(L)

- Plans
  - Structure
    - *Event – Context – Body*
  - Contents
    - *Actions*
    - *Internal Actions*
    - *Achievement Goals*
    - *Test Goals*
    - *Mental Notes*
    - *Expressions*
  - Example

```
+!start :: light(green) <- !action1;!action2.
```

IMPLEMENTED FEATURES

- Strong Negation

  `~like(cakes)` **(different use from:** `not(like(cakes))` **)**

- Belief Annotations

  ```
  light(green)#[source(agent26)]
  ```

- Plan Annotations

  ```
  +light(green)#[source(agent3)]:: true <- !action.
  ```

- Complicated Mental Rules

  ```
  findAllTemp(List):-
     findall(X,temp(X),List1) &
     sort(List1,List).
  ```

IMPLEMENTATION (OUTLINE)

- **Parser**
- **Solver**
- **Design of the Meta-Interpreter**
- **Failure Handling**

IMPLEMENTATION (PARSER)

- *The parser loads the program, by creating a set of Prolog facts for every program statement it parses, using a failure driven loop.*
- `Belief(B,Atts)`
  - `object(front)#[source(agent1)]`
  - `object(front)#[source(agent2)]`
  - `object(front)#[alertness(high)]`

  } `belief(B,Atts)`

`B: object(front)`

`Atts: [source(agent1),source(agent2),alertness(high)]`

IMPLEMENTATION (PARSER)

- `Rule(Head,Body)`
  - `Head :- B1 & B2` → `rule(Head,B1 & B2)`

  `Head: over_limit(V)`

  `Body: limit(L) & V>=L`

- `Plan(Event,Context,Body)`
  - `Event :: Context <- Body` → `plan(Event,Context,Body)`

  `Event: +!speak`

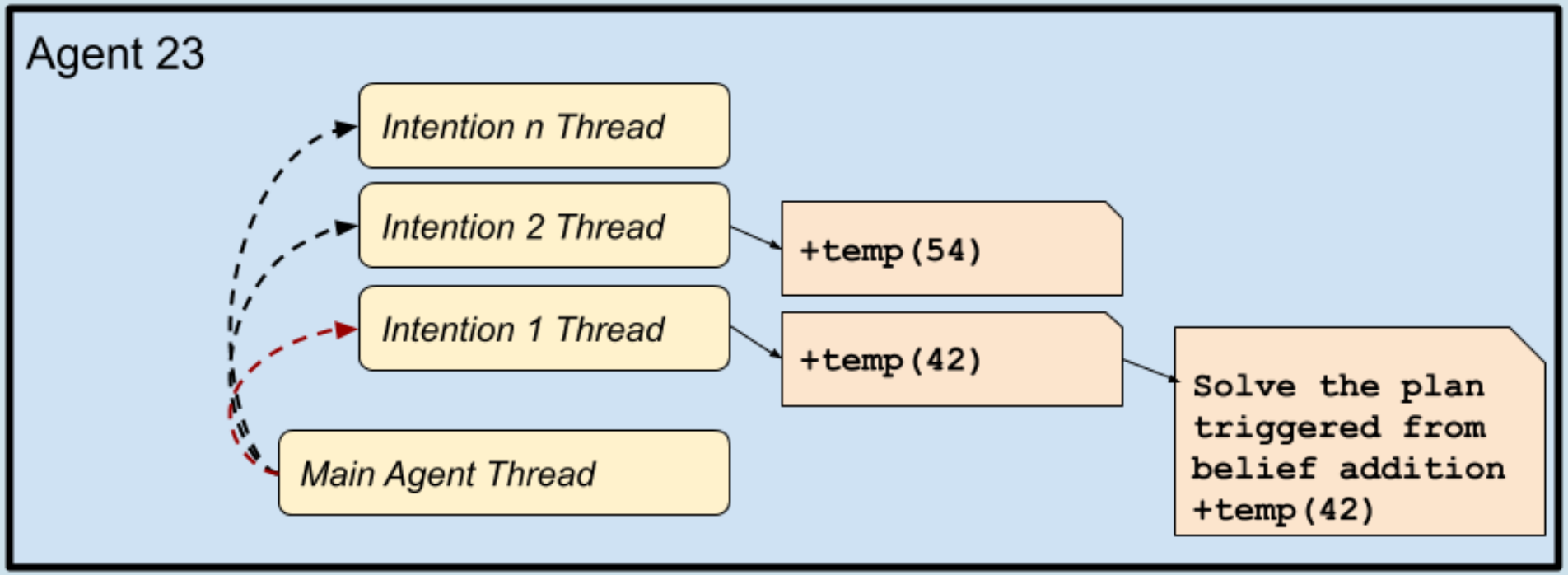
  `Context: mood(nice)`

  `Body: !say_hello`

IMPLEMENTATION (SOLVER)

- Achievement and Test Goals (`+!g, +?g`)
- Goals pursued as separate intentions (`^^g`)
- Belief Addition/Deletion Goals (`+b, -b`)
- Plans triggered by Belief Addition/Deletion (`+/-b::true<-actions`)
- Expressions/Prolog Predicates

DESIGN OF THE META-INTERPRETER



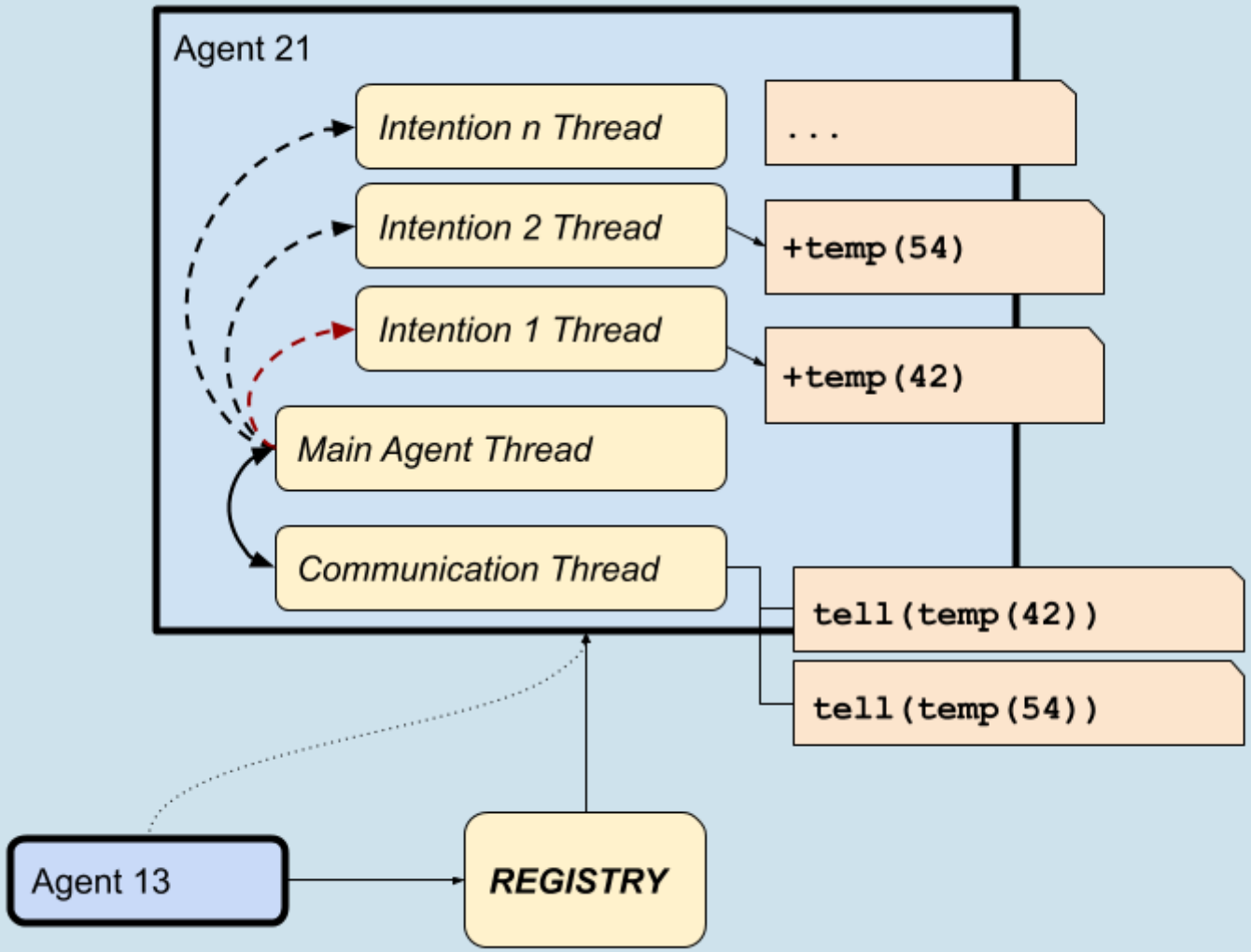
DESIGN OF THE META-INTERPRETER

- New Intention on:
  - Plan Execution
  - Goal ^^g
  - External Belief Addition/Deletion
  - Internal Belief Addition/Deletion

IMPLEMENTATION (FAILURE HANDLING)

Plan Executed	Description	Recovery
`+g`	Plan found and executed with success	✓
`-g#[error(non_applicable_plan)]`	No applicable plan found, but an annotated plan for that case was provided and executed	✓
`-g#[error(no_relvevant_plan)]`	No relevant plan found, but an annotated plan for that case was provided and executed	✓
`-g`	A (known or unknown) failure occurred, but an annotated plan was provided and executed	✓
`-g`	Failure handling plan found but it is non-applicable	✗
`---`	No applicable plan found	✗
`---`	No relevant plan found	✗

AGENTS' COMMUNICATION (DESIGN)



AGENTS' COMMUNICATION (MESSAGE TYPES)

`tell(Content)`

- *Adds a belief in receiver's belief base*

`untell(Content)`

- *Removes a belief from receiver's belief base*

`achieve(Content)`

- *Asking the receiver for the execution of an achievement plan*

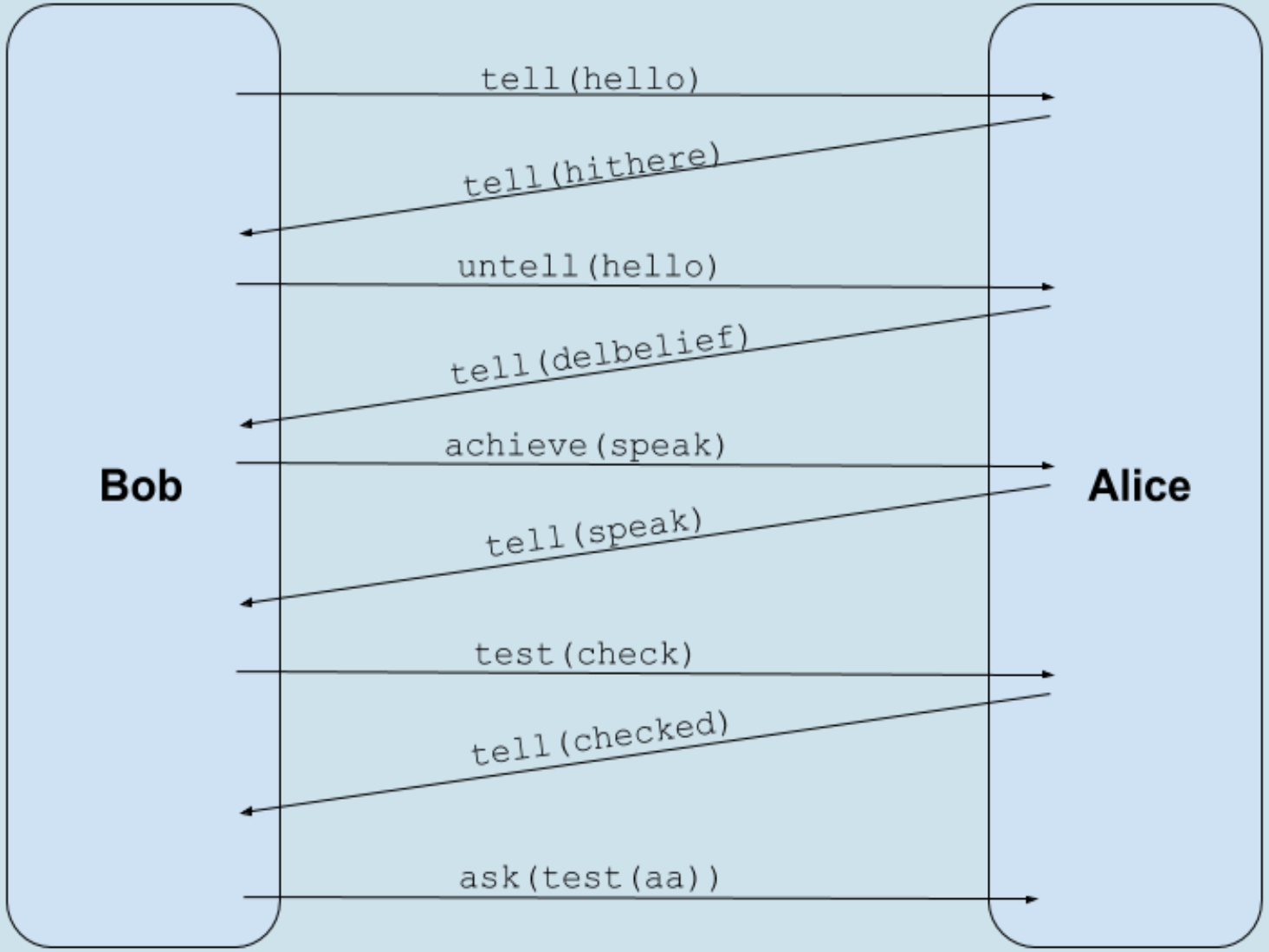
`test(Content)`

- *Asking the receiver for the execution of a test plan*

`ask(Content)`

- *Asking the receiver if a condition is true or not*

TEST CASES


Bob
Alice
tell(hello)
tell(hithere)
untell(hello)
tell(delbelief)
achieve(speak)
tell(speak)
test(check)
tell(checked)
ask(test(aa))

CONCLUSIONS

- Combining logic with agent oriented programming features
  - Strong Negation
  - Belief & Plan Annotations
  - Complicated Mental Rules
  - Higher-order Prolog predicates
  - Failure Handling mechanism
- Multithreaded Application
  - Autonomous Agent's Intention Handling
  - Communication Design
- Asynchronous Communication

FUTURE WORK

- CLP Implementation
- Planning from 1st Principals
- Semantics' Extension
- Failure Handling mechanism Extension
- Synchronous Communication Implementation
- Debugging Techniques Investigation
- Testing in truly distributed Cloud environments

THANK YOU FOR YOUR ATTENTION

A Prolog Meta-Interpreter for AgentSpeak(L)

M.Sc. THESIS

of

Dimitrios Dimitriadis

Thessaloniki, 21 June 2019