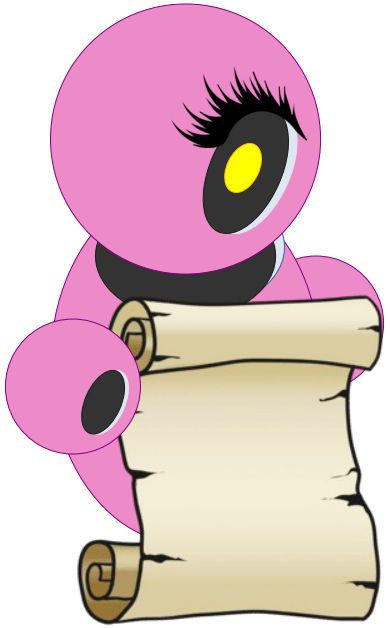
Introduction to Distributed and Embedded Multi-agent Systems

Carlos Eduardo Pantoja[1]
Nilson Mori Lazarin[1,2]

1. Centro Federal de Educação Tecnológica (CEFET/RJ) - 2. Universidade Federal Fluminense (UFF), Brasil

Março, 2024

BELIEFS AND RULES



Jason Framework

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

Jason Framework

Jason | a Java-based interpreter for an extended version of AgentSpeak.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

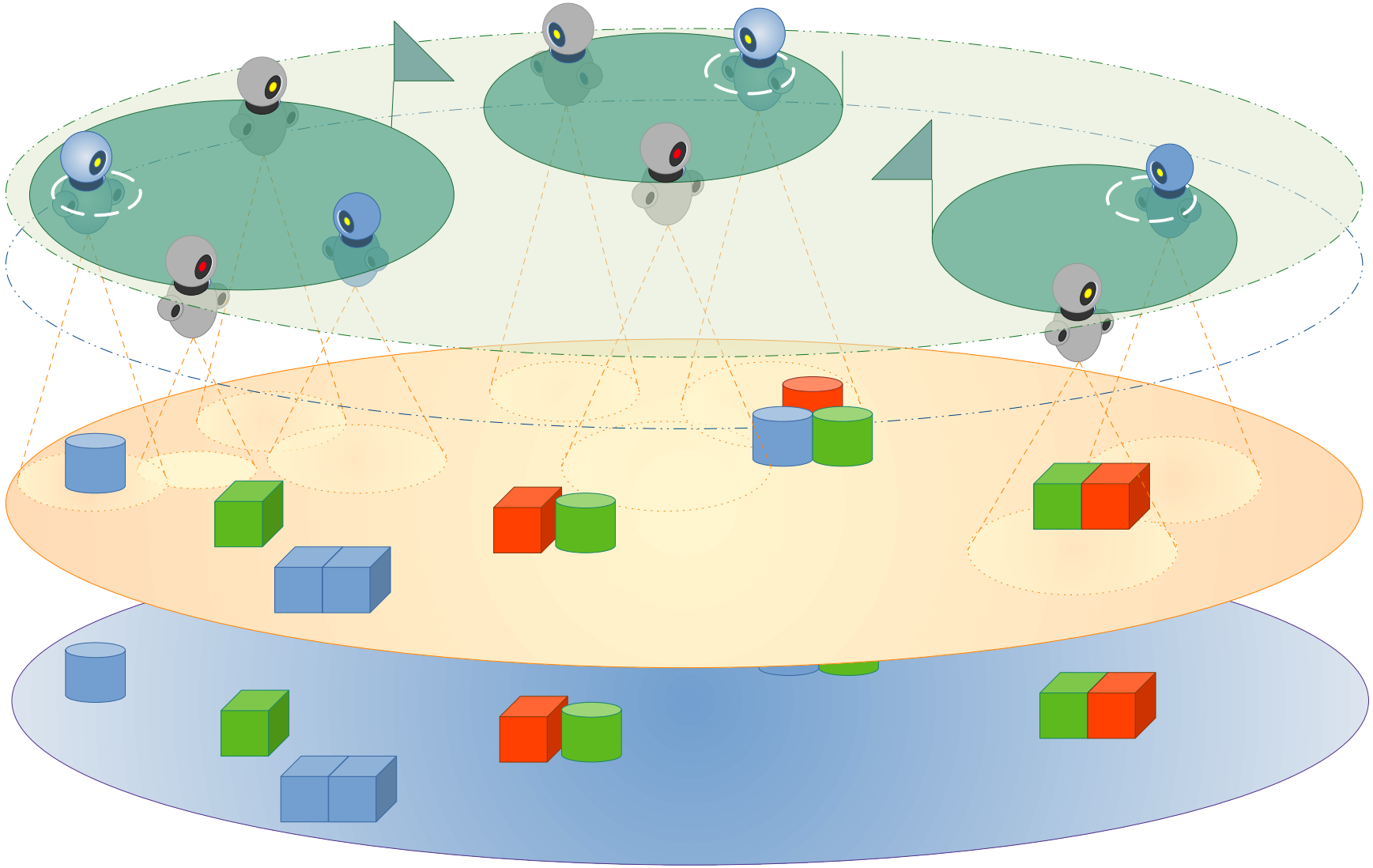
Jason Framework

Jason | a Java-based interpreter for an extended version of AgentSpeak.

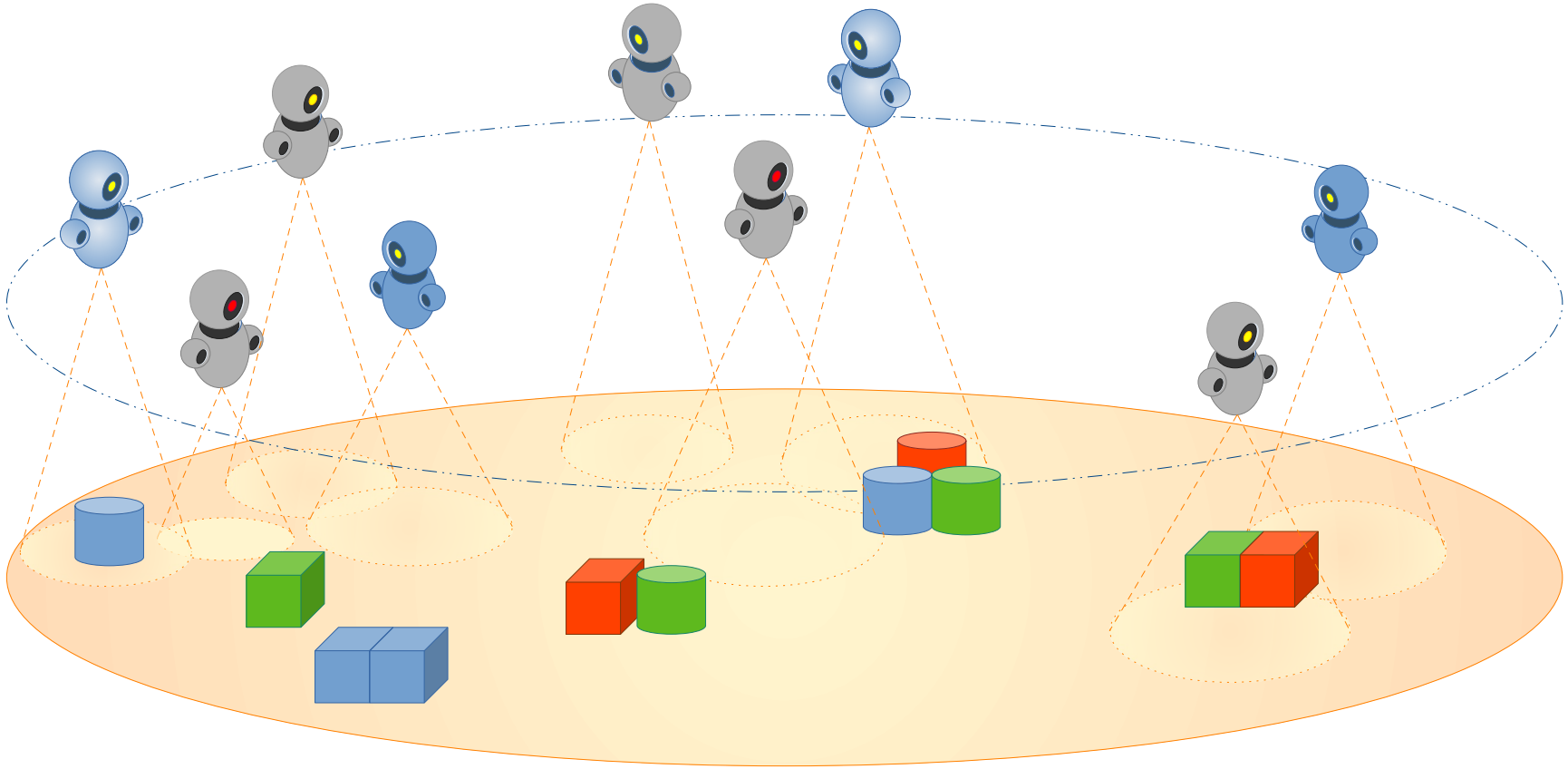
http://jason.sf.net

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

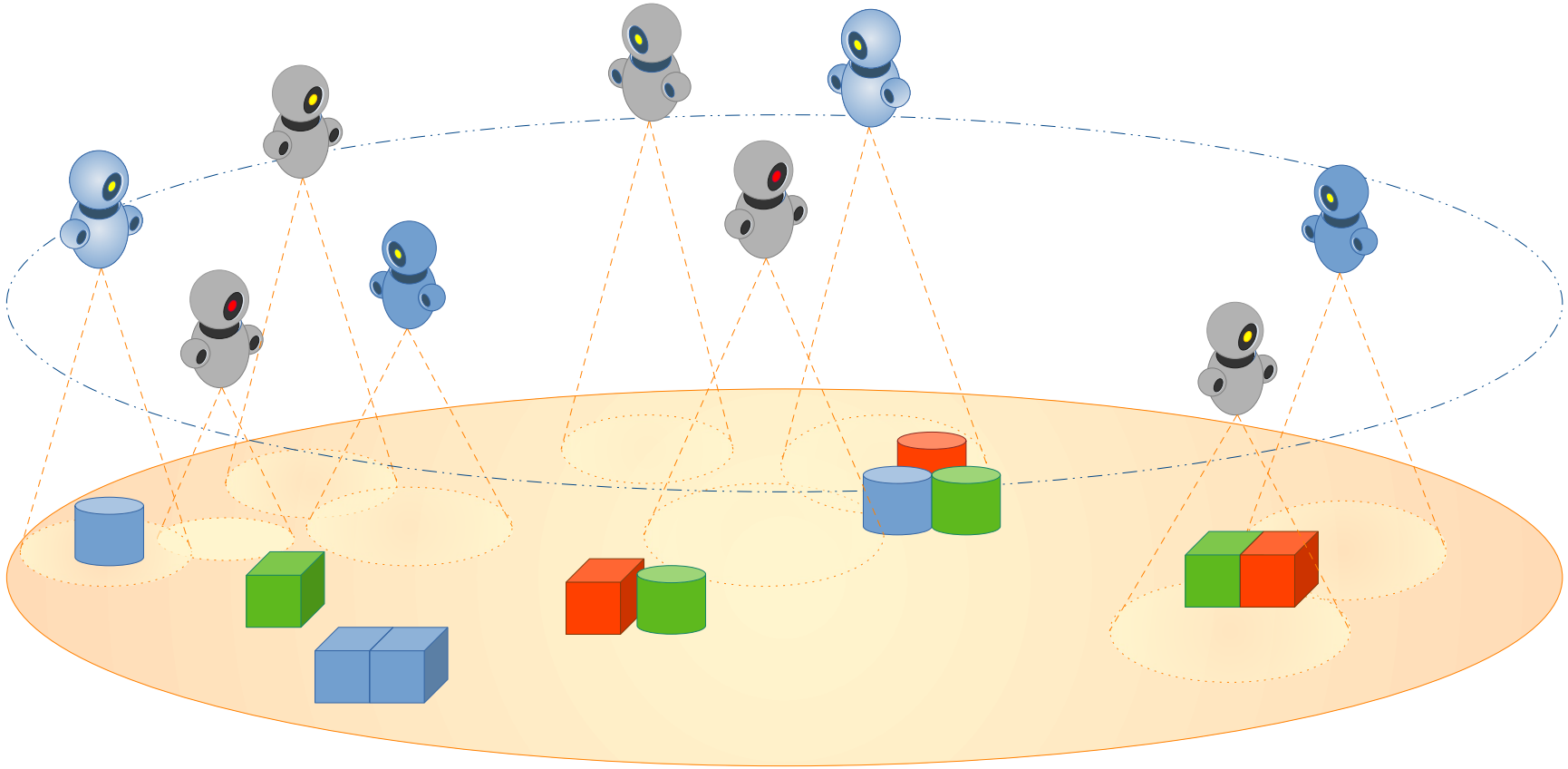
Jason Framework



Jason Framework



Jason Framework



Using Jason, it is possible to program the **agent** and the **endogenous environment** dimensions.

Jason Framework: Crenças

Em Jason, um agente armazena as **informações** percebidas do ambiente; as informações internas; e informações de comunicação através de crenças.

Jason Framework: Crenças

Em Jason, um agente armazena as **informações** percebidas do ambiente; as informações internas; e informações de comunicação através de crenças.

As **crenças** são armazenadas em uma **Base de Crenças** (Belief Base).

Jason Framework: Crenças

Em Jason, um agente armazena as **informações** percebidas do ambiente; as informações internas; e informações de comunicação através de crenças.

As **crenças** são armazenadas em uma **Base de Crenças** (Belief Base).

As crenças são representadas como predicados da **lógica tradicional**. Os **predicados** representam propriedades particulares.

Jason Framework: Beliefs

1. **Mental Notes (self)**
   - Informações adicionadas na base de crenças pelo próprio agente.

Jason Framework: Beliefs Types

1. **Mental Notes (self)**
   - Informações adicionadas na base de crenças pelo próprio agente.

2. **Communication (agent)**
   - Informações obtidas pelo agente através da interação com outros agentes.

Jason Framework: Beliefs Types

1. **Mental Notes (self)**
   - Informações adicionadas na base de crenças pelo próprio agente.

2. **Communication (agent)**
   - Informações obtidas pelo agente através da interação com outros agentes.

3. **Environmental Perceptions (percepts)**
   - Informações coletadas pelo agente que são relativas ao sensoriamento constante do ambiente.

Beliefs: Format

belief [source(type)]

Beliefs: Format

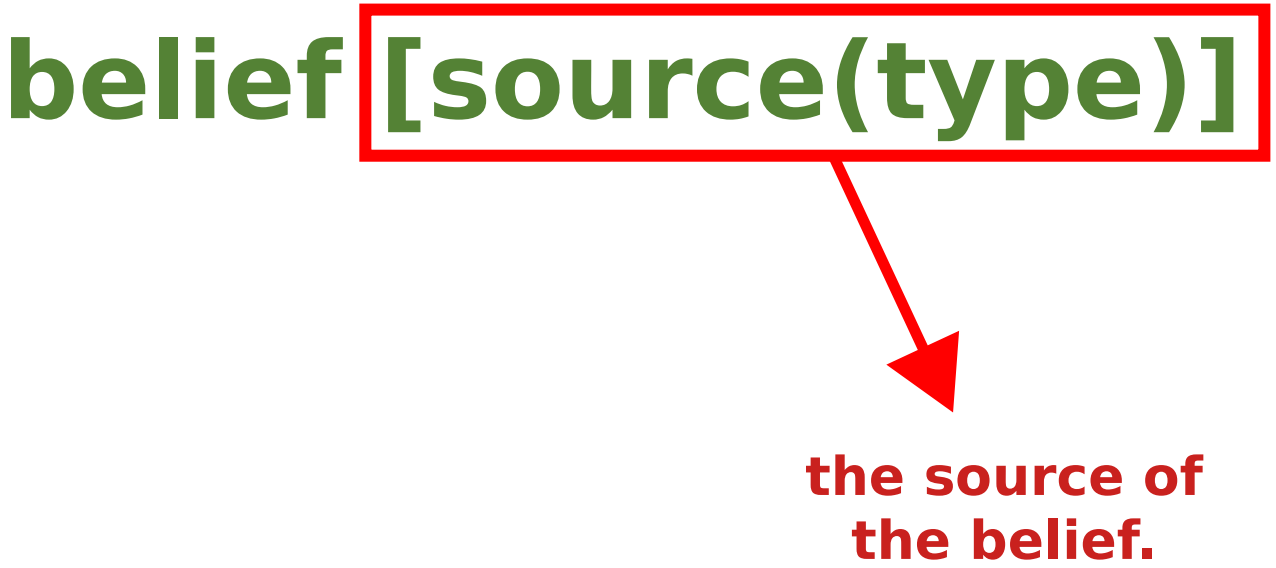
belief [source(type)]

Beliefs: Format

Beliefs: Format

belief [source(type)]

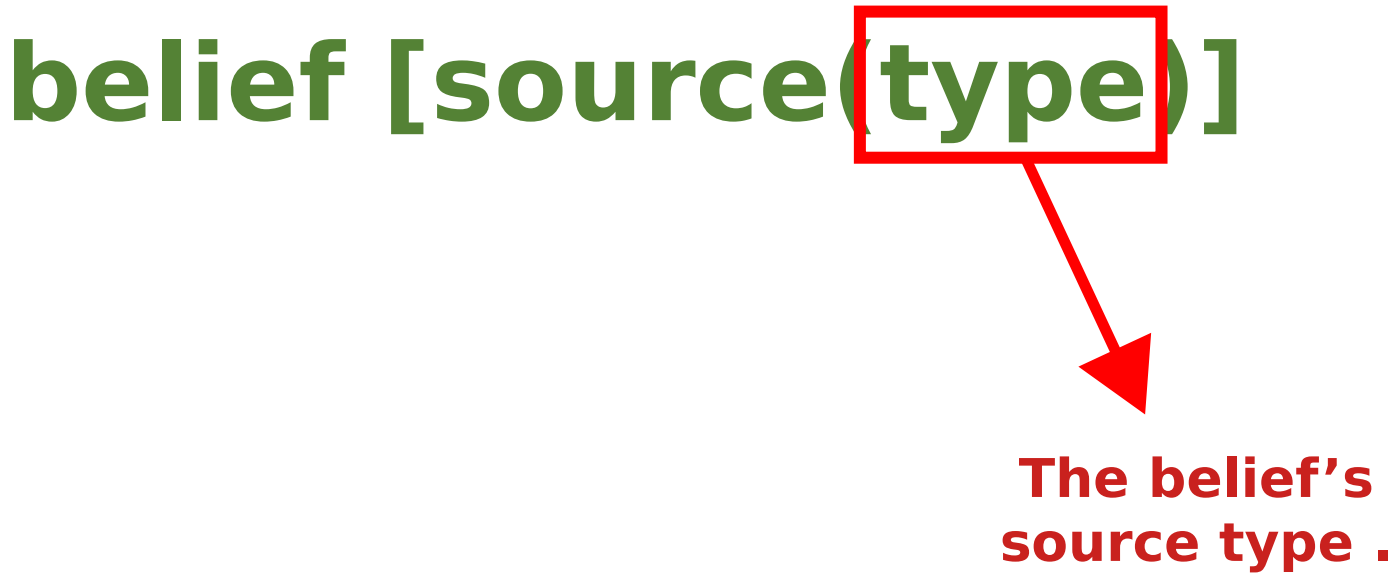
Beliefs: Format



Beliefs: Format

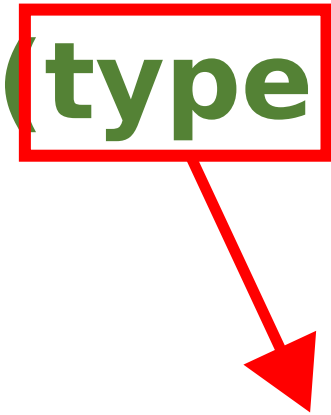
belief [source(type)]

Beliefs: Format



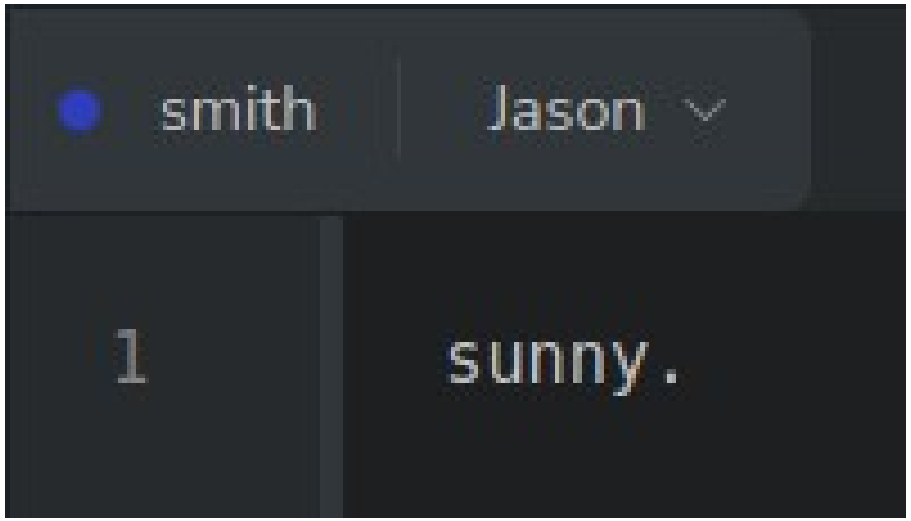
Beliefs: Format

belief [source(type)]

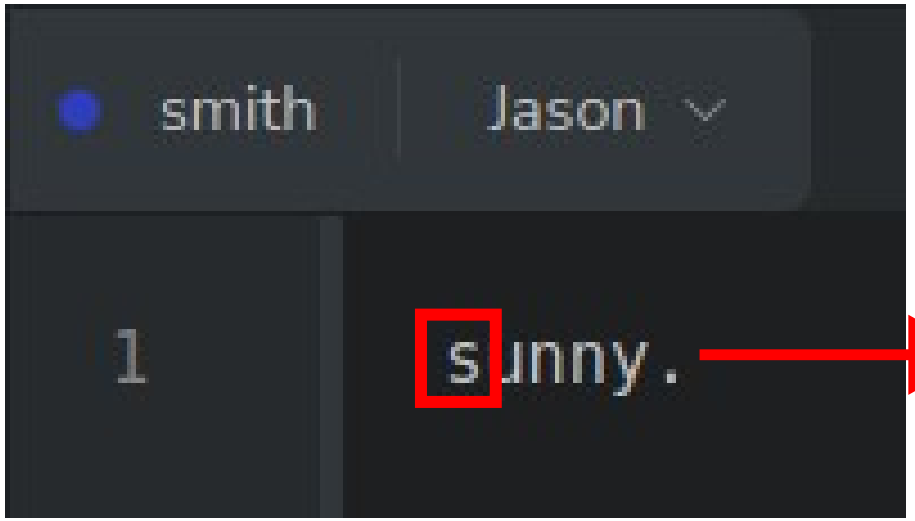


It can be self, percepts, or from other agents.

Initial Beliefs: Ment

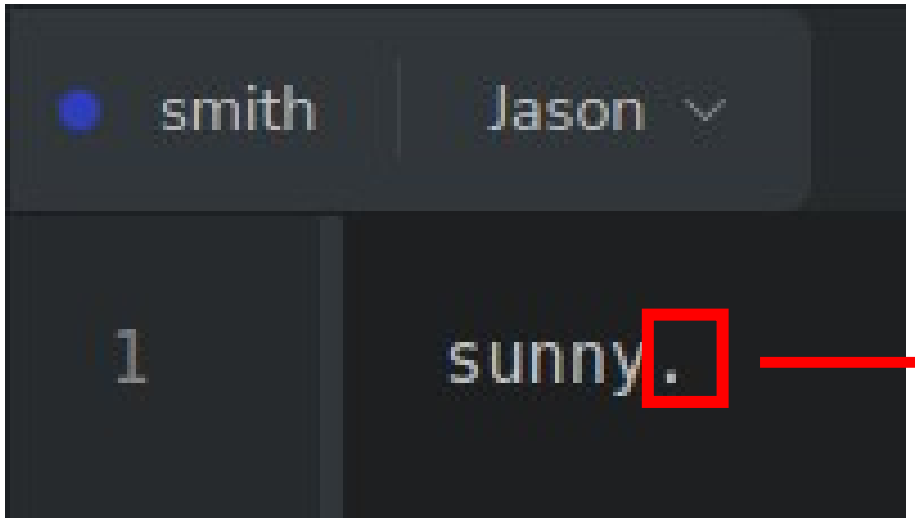


Initial Beliefs: Mental Notes



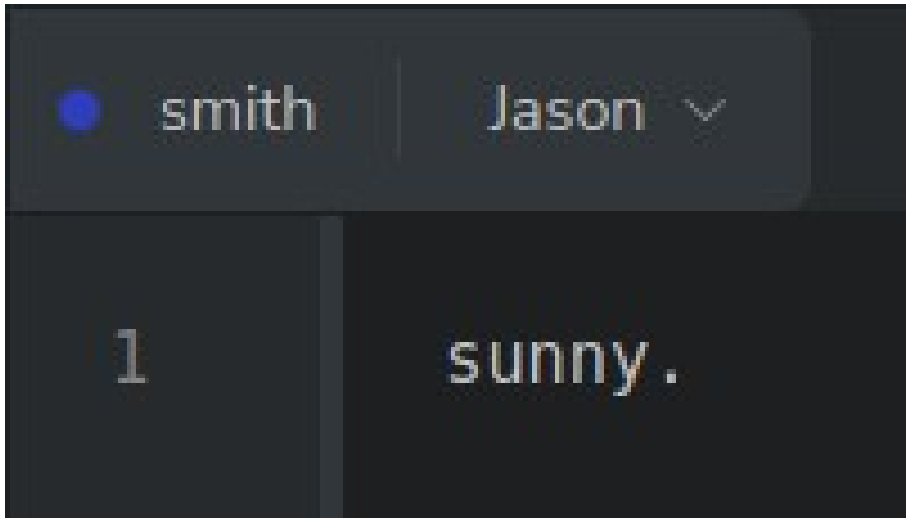
every belief must start with a **lowercase** letter.

Initial Beliefs: Mental Notes

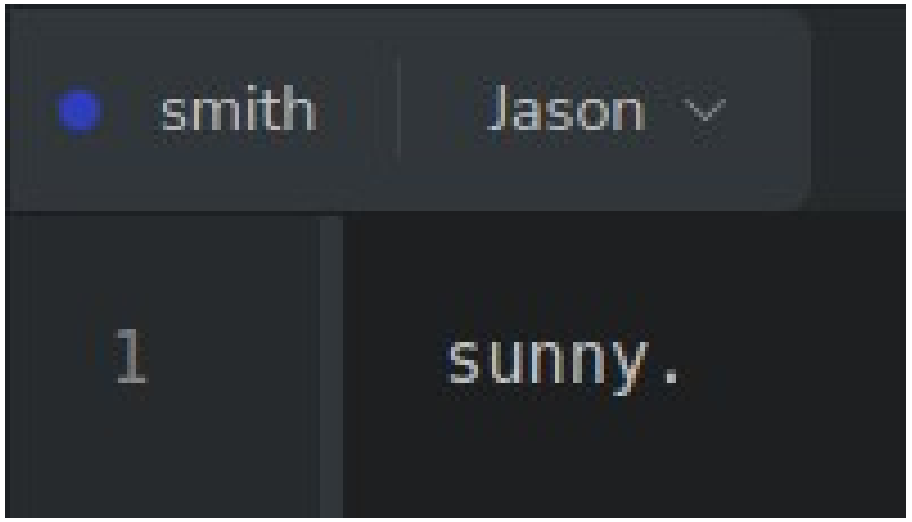


and it must end with a **period**.

Initial Beliefs: Mental Notes



Initial Beliefs: Mental Notes

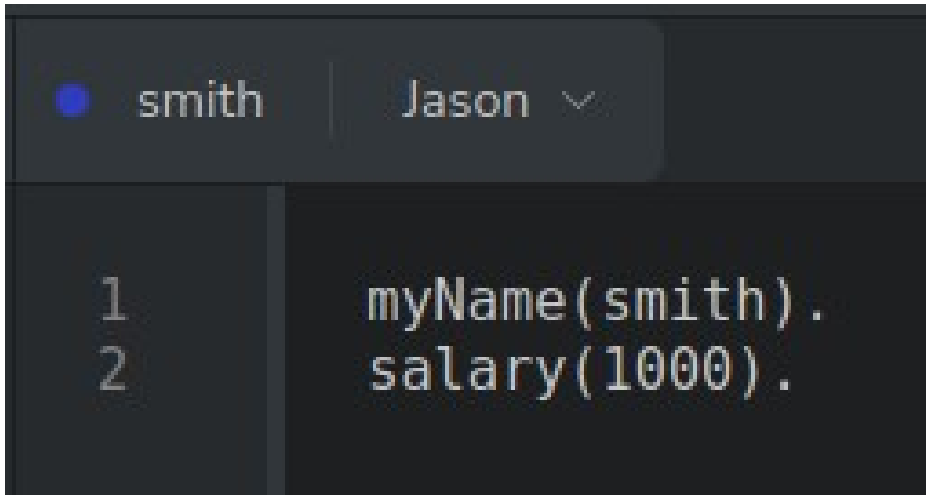


Inspection of agent **smith**

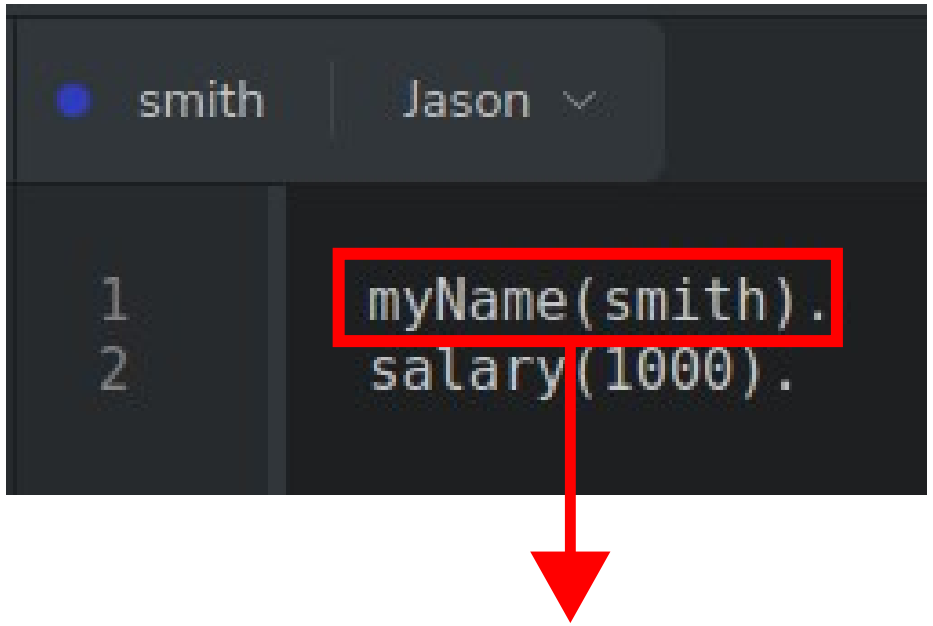
- **Beliefs** sunny[source(self)].

- **Annotations**

Initial Beliefs: Mental Notes

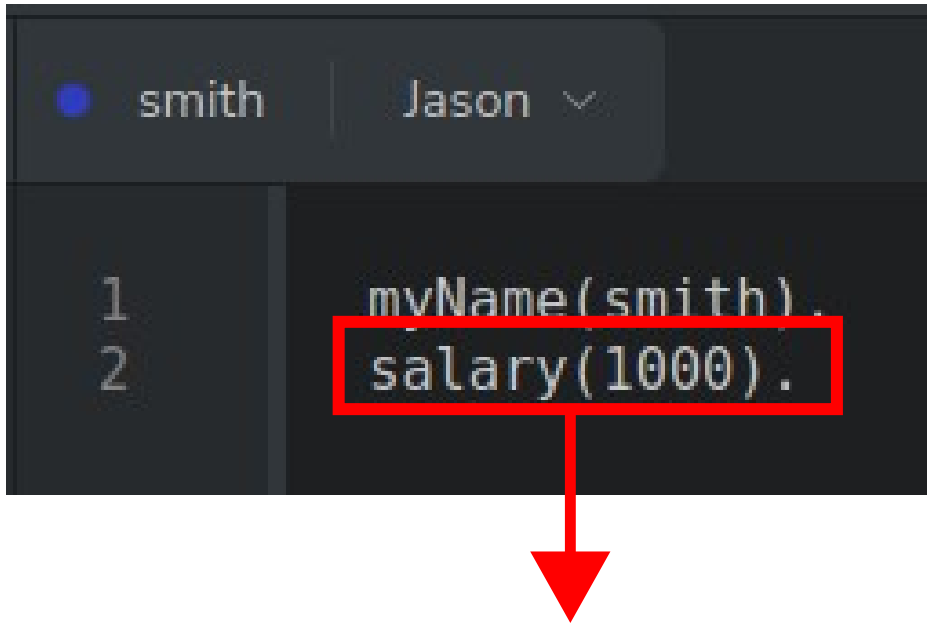


Initial Beliefs: Mental Notes



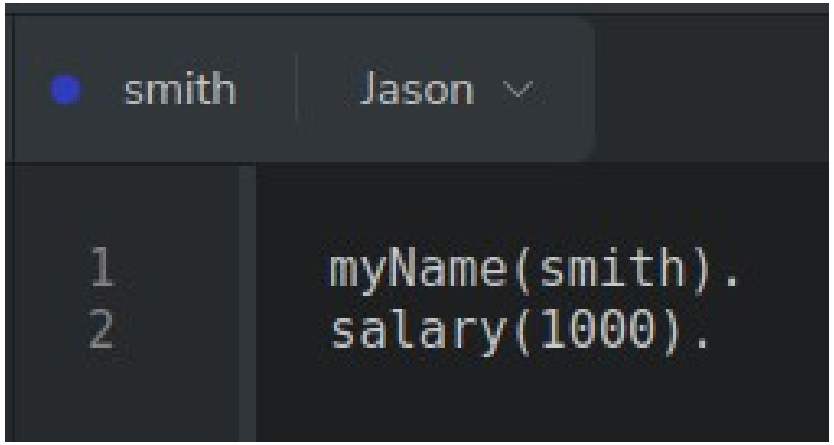
predicate(predicate)

Initial Beliefs: Mental Notes

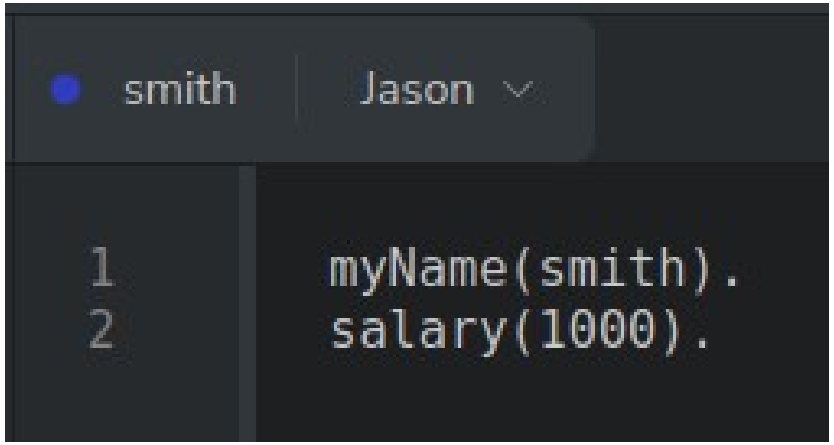


predicate(int)

Initial Beliefs: Mental Notes



Initial Beliefs: Mental Notes



Inspection of agent **smith**

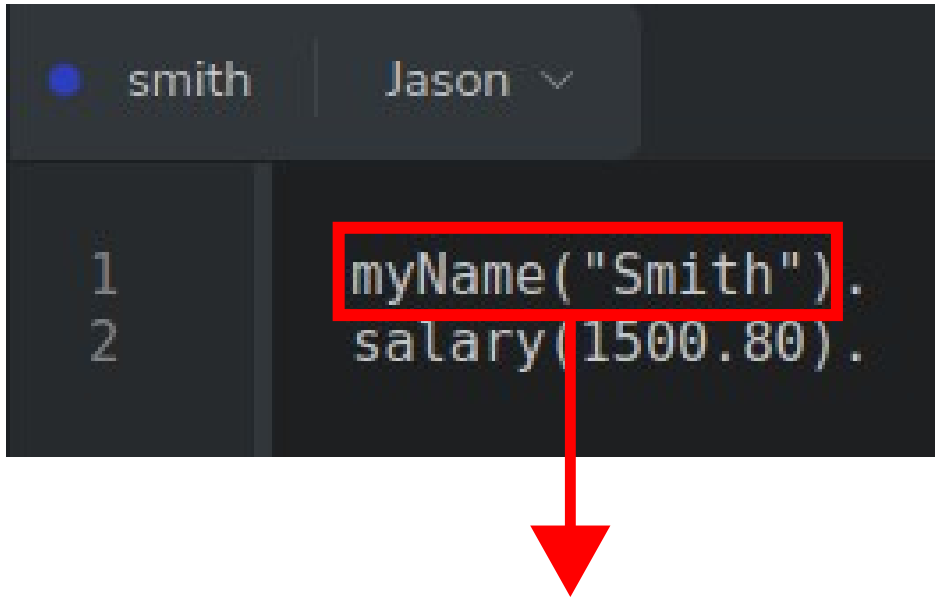
- **Beliefs**
  - myName(smith)[source(self)].
  - salary(*1000*)[source(self)].
- **Annotations**

Initial Beliefs: Mental Notes

smith Jason

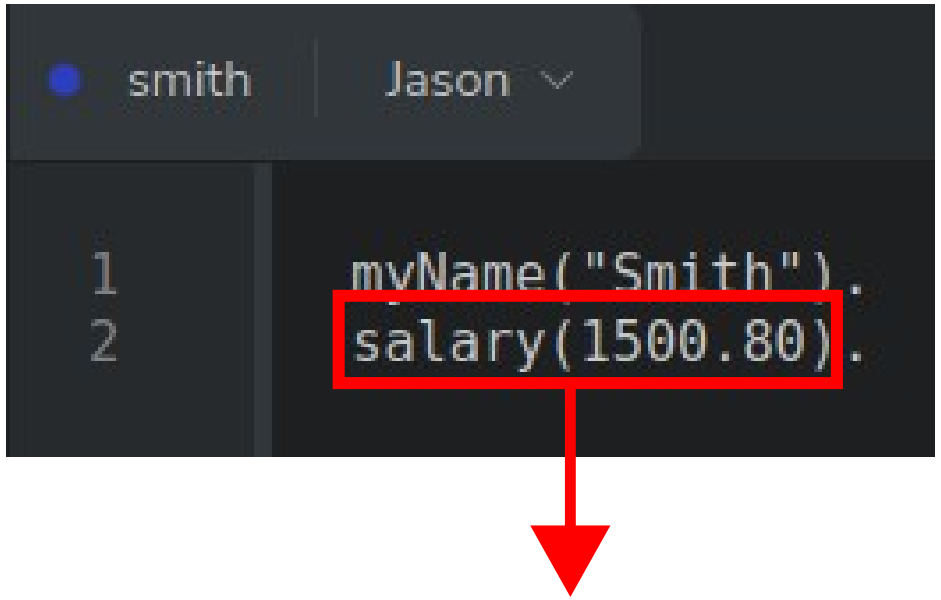
```
1    myName("Smith").
2    salary(1500.80).
```

Initial Beliefs: Mental Notes



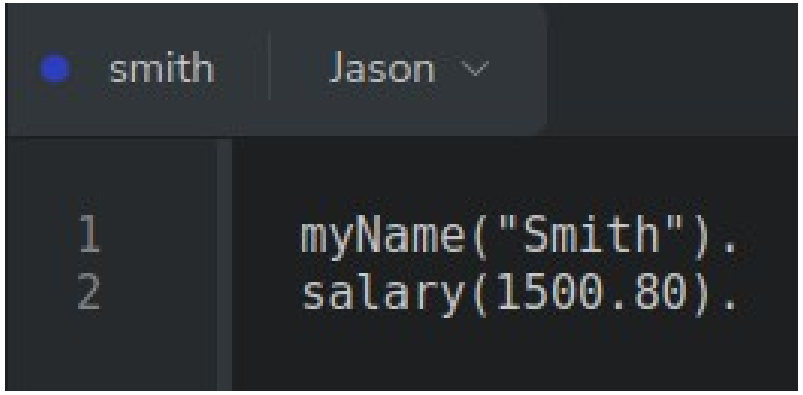
predicate(String)

Initial Beliefs: Mental Notes

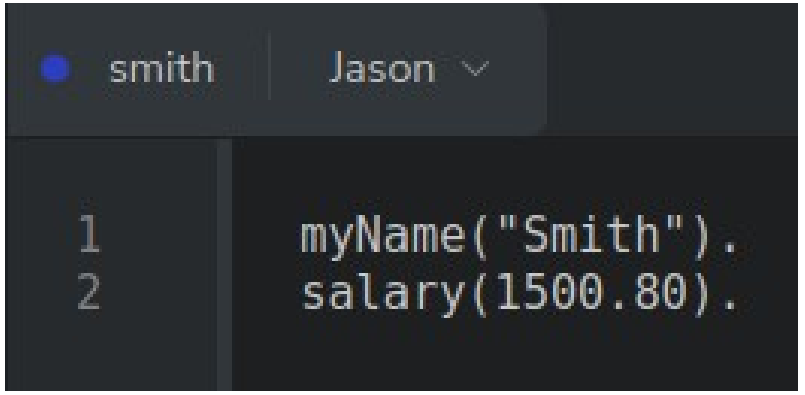


predicate(float)

Initial Beliefs: Mental Notes



Initial Beliefs: Mental Notes

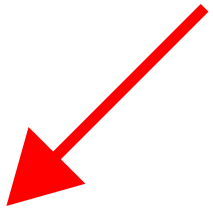


Beliefs: Format

predicate(value)

Beliefs: Format

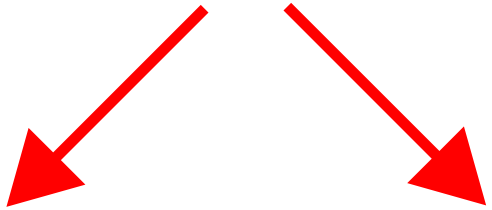
predicate(value)



predicate

Beliefs: Format

predicate(value)



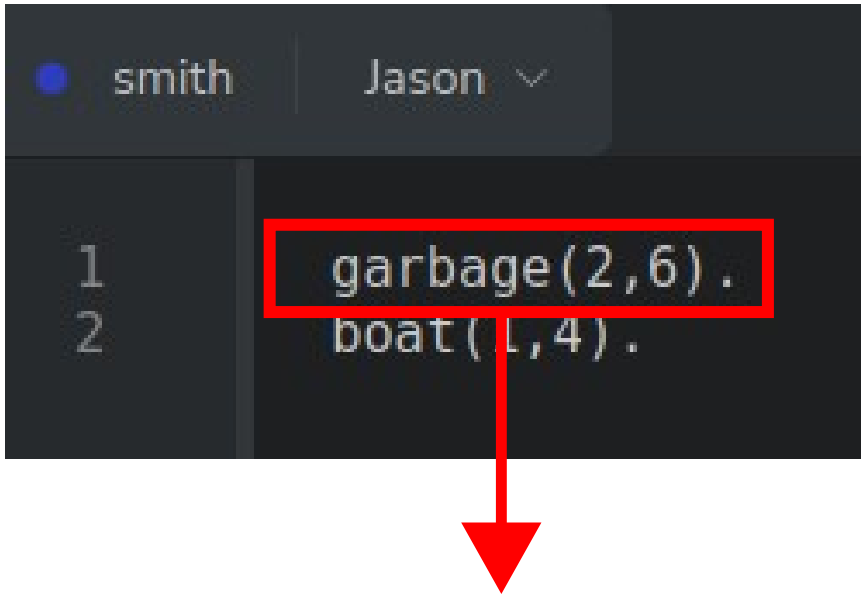
predicate int, float, String, etc.

Initial Beliefs: Mental Notes

smith Jason

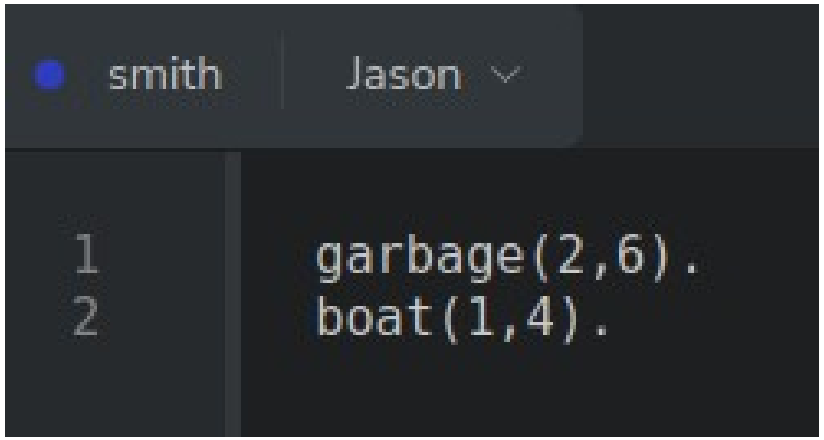
```
1  garbage(2,6).
2  boat(1,4).
```

Initial Beliefs: Mental Notes

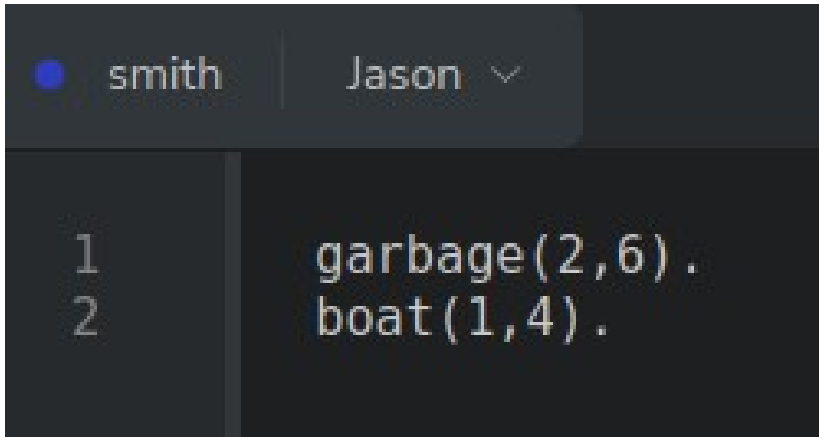


predicate(int, int)

Initial Beliefs: Mental Notes



Initial Beliefs: Mental Notes



Inspection of agent **smith**

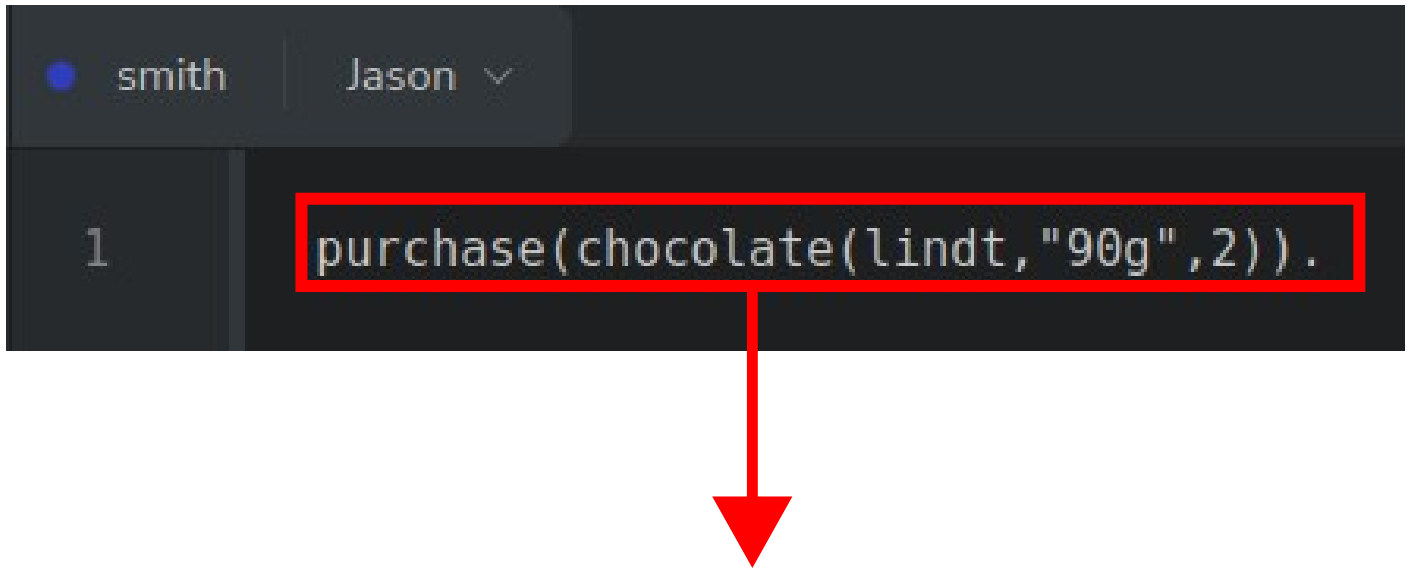
- **Beliefs**
  - boat(*1*,*4*)[source(self)].
  - garbage(*2*,*6*)[source(self)].
- **Annotations**

Initial Beliefs: Mental Notes

smith Jason

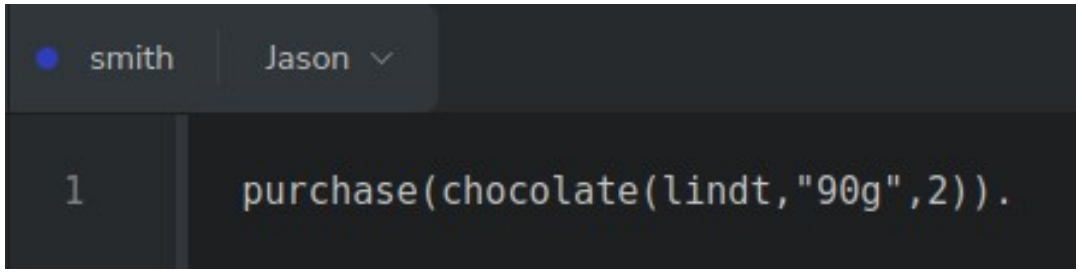
```
1    purchase(chocolate(lindt,"90g",2)).
```

Initial Beliefs: Mental Notes



predicate(predicate(predicate, String, int)

Initial Beliefs: Mental Notes



Initial Beliefs: Mental Notes

smith | Jason

```
1   purchase(chocolate(lindt,"90g",2)).
```

Inspection of agent **smith**

- Beliefs	purchase(chocolate(lindt,"90g",2))[source(self)].
- Annotations	

Beliefs: Format

predicate(predicate)

Beliefs: Format

predicate(predicate)

Beliefs: Format

predicate(predicate(predicate))

Beliefs: Format

predicate(predicate(predicate))

Beliefs: Format

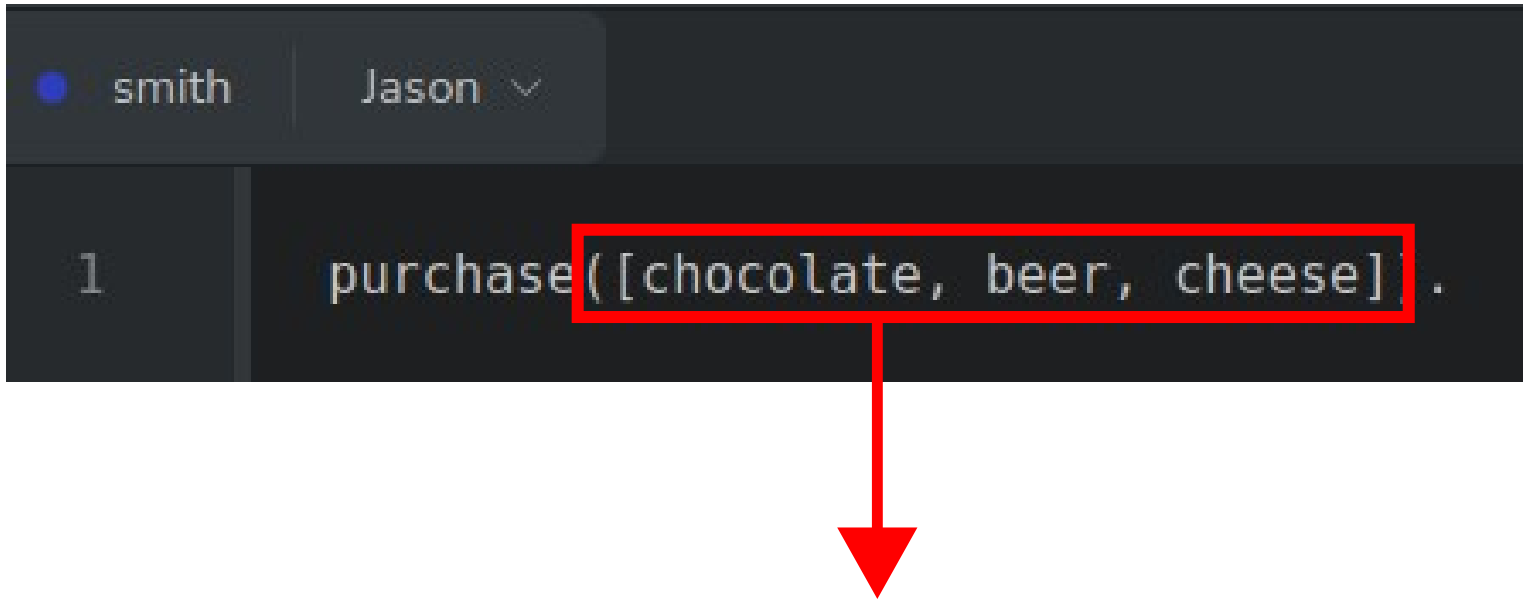
predicate(predicate(predicate(...)))

Initial Beliefs: Mental Notes

smith Jason

```
1    purchase([chocolate, beer, cheese]).
```

Initial Beliefs: Mental Notes



predicate([predicate, predicate, predicate])

Initial Beliefs: Mental Notes

smith Jason

```
1   purchase([chocolate, beer, cheese]).
```

Initial Beliefs: Mental Notes

smith | Jason

```
1   purchase([chocolate, beer, cheese]).
```

Inspection of agent **smith**

- Beliefs	purchase([chocolate,beer,cheese])[source(self)].
- Annotations	

Beliefs: Format

predicate([list])

Beliefs: Format

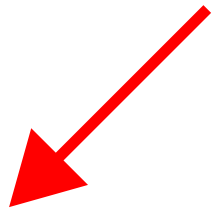
predicate([list])

Beliefs: Format

[value, value, ...]

Beliefs: Format

[value, value, ...]



predicate

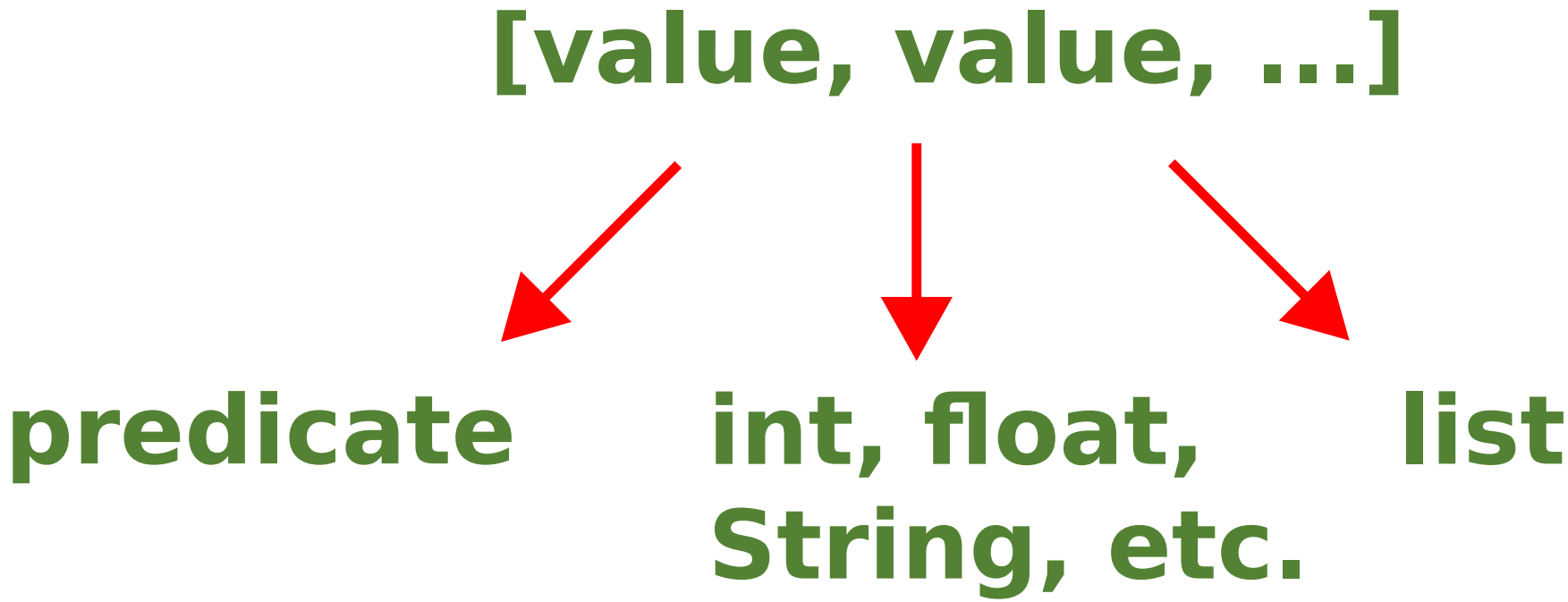
Beliefs: Format

[value, value, ...]

predicate

int, float,
String, etc.

Beliefs: Format



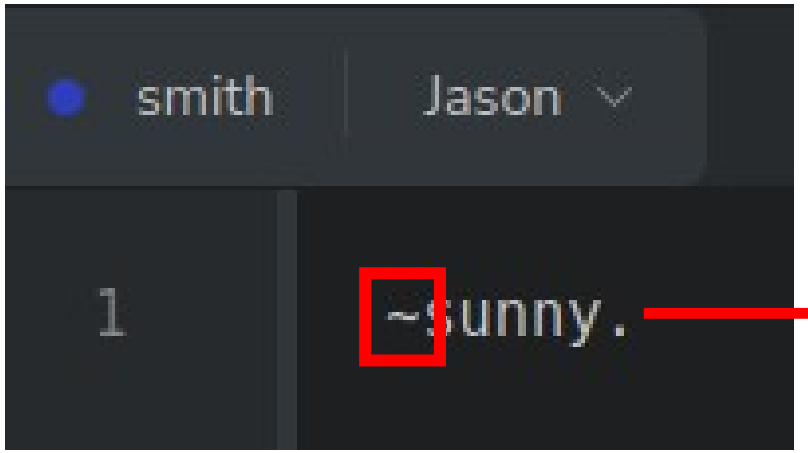
Beliefs: Strong Negation

~belief [source(type)]

Beliefs: Strong Negation

every strong negation starts with ~.

Initial Beliefs



it negates the predicate.

Initial Beliefs: Mental Notes

Initial Beliefs: Mental Notes

Inspection of agent **smith**

- **Beliefs** ~sunny[source(self)].
- **Annotations**

Jason Framework: Rules

A **rule** is a statement that defines relationships between facts (beliefs).

Jason Framework: Rules

A **rule** is a statement that defines relationships between facts (beliefs).

The rule operator can be understood as a **logical implication**. It reads as **"if"** or **"is true when"**.

Rules: Format

head :- body.

Rules: Format

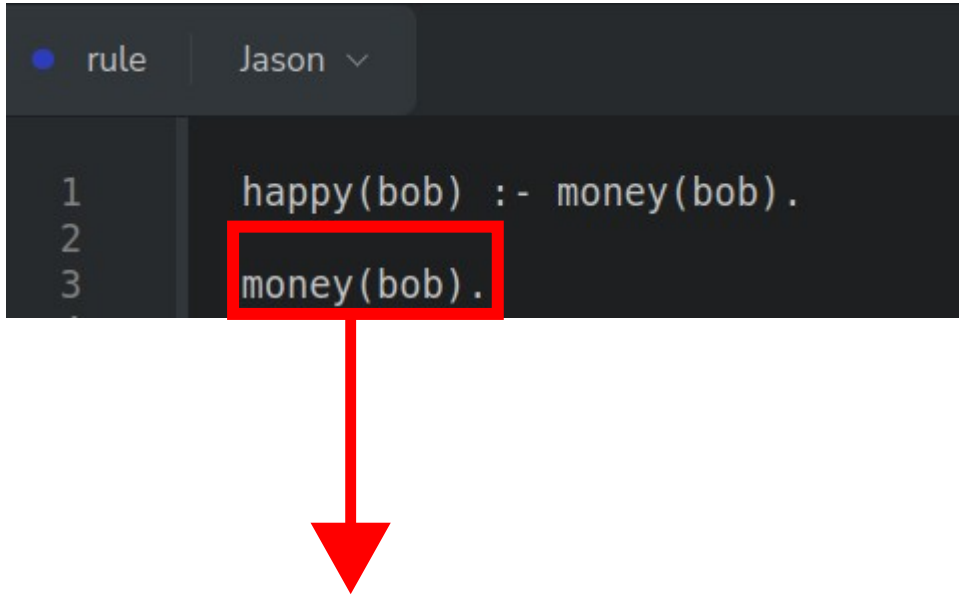
Rules: Format

Rules: Predicate

rule | Jason

```
1  happy(bob) :- money(bob).
2
3  money(bob).
```

Rules: Predicate



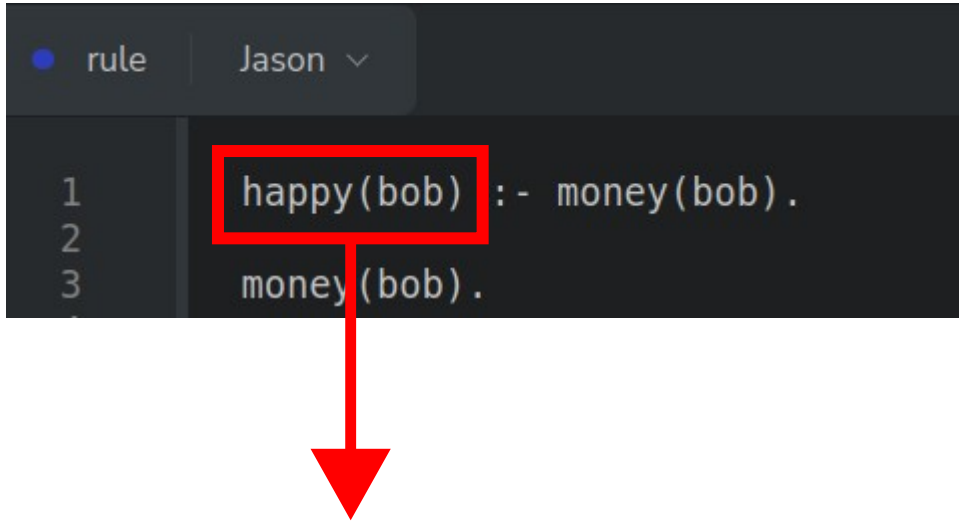
Rules: Predicate

rule | Jason

```
1  happy(bob) :- money(bob).
2
3  money(bob).
```

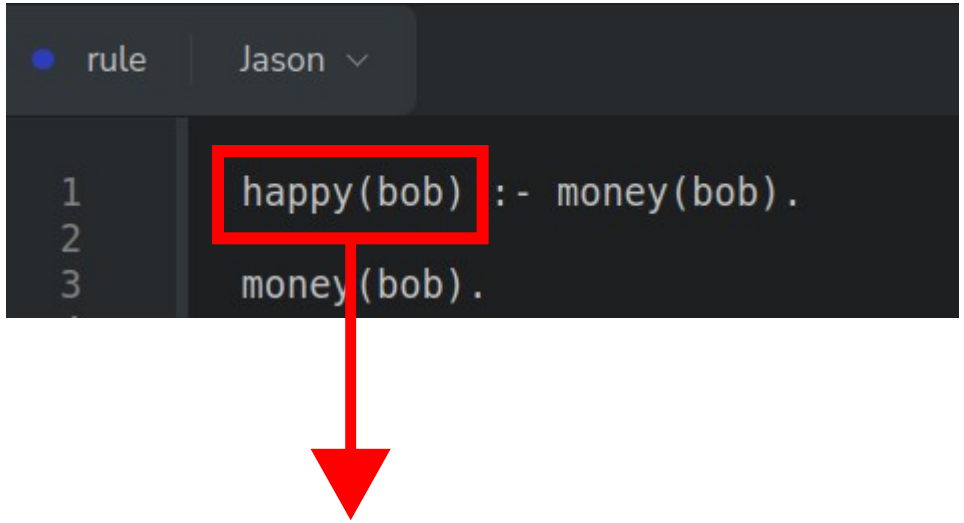
... this condition is satisfied.

Rules: Predicate



So, it holds ...

Rules: Predicate



... but it is NOT A BELIEF.

Rules: Predicate

```
rule    Jason
1   happy(bob) :- money(bob).
2
3   money(bob).
```

Inspection of agent **rule**

- Beliefs	money(bob)[source(self)].
- Rules	happy(bob) :- money(bob).
- Annotations	

Jason Framework: Variables

In Jason, a variable is a **placeholder** for an **unknown value**.

Jason Framework: Variables

In Jason, a variable is a **placeholder** for an **unknown value**.

- denoted by a string consisting of an uppercase letter or an underscore;

Jason Framework: Variables

In Jason, a variable is a **placeholder** for an **unknown value**.

- denoted by a string consisting of an uppercase letter or an underscore;
- it must be instantiated with specific values while executing the agent's reasoning;

Jason Framework: Variables

In Jason, a variable is a **placeholder** for an **unknown value**.

- denoted by a string consisting of an uppercase letter or an underscore;
- it must be instantiated with specific values while executing the agent's reasoning;
- The underscore is a special variable that is often used when the value of a variable is not needed or not used. It is sometimes referred to as a **"don't care"** variable.

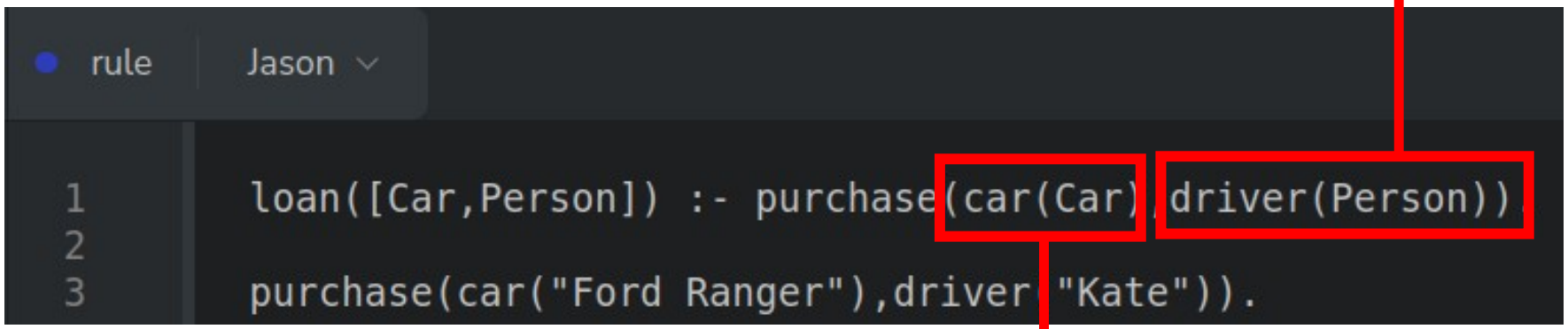
Rules: With Variables

rule Jason

```
1  loan([Car,Person]) :- purchase(car(Car),driver(Person)).
2
3  purchase(car("Ford Ranger"),driver("Kate")).
```

Rules: With Variables

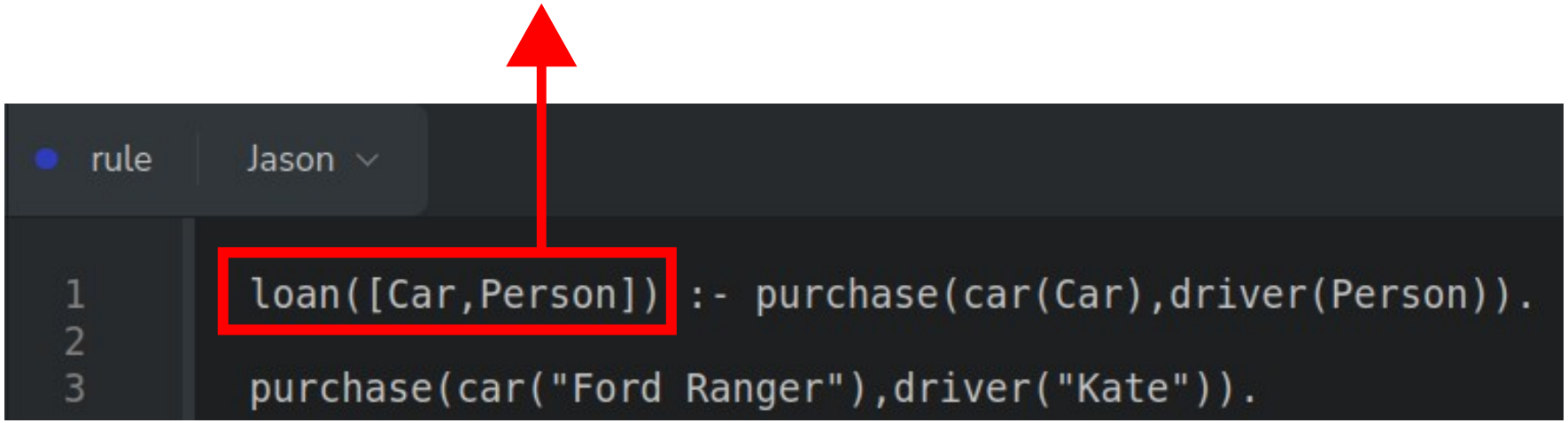
predicate(Variable)



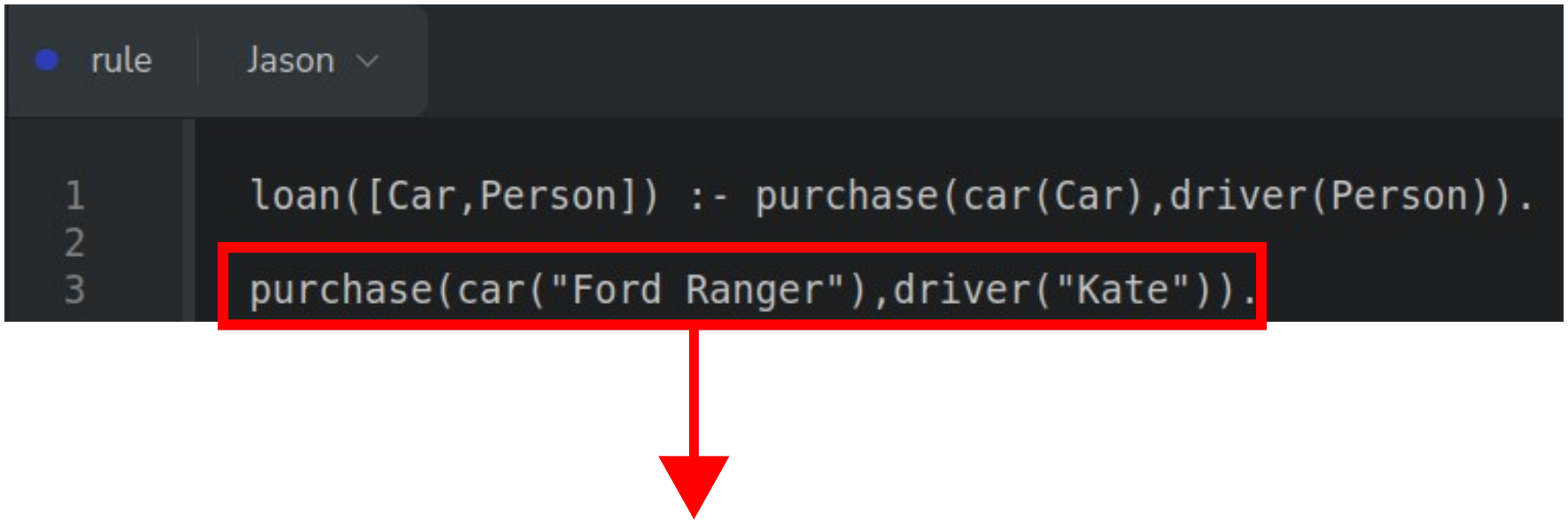
predicate(Variable)

Rules: With Variables

predicate([Variable, Variable])



Rules: With Variables



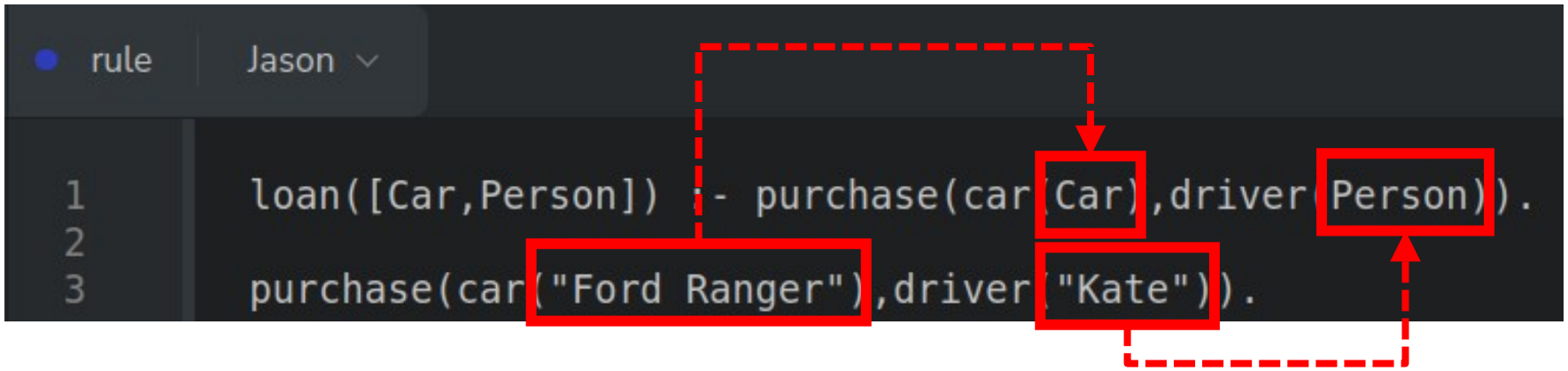
If a belief exists...

Rules: With Variables

rule Jason

```
1  loan([Car,Person]) :- purchase(car(Car),driver(Person)).
2
3  purchase(car("Ford Ranger"),driver("Kate")).
```

Rules: With Variables



... it binds the value to a variable (unification).

Rules: With Variables

rule Jason

```
1  loan([Car,Person]) :- purchase(car(Car),driver(Person)).
2
3  purchase(car("Ford Ranger"),driver("Kate")).
```

Rules: With Variables

rule | Jason

```
1   loan([Car,Person]) :- purchase(car(Car),driver(Person)).
2
3   purchase(car("Ford Ranger"),driver("Kate")).
```

Inspection of agent **rule**

- Beliefs	purchase(car("Ford Ranger"),driver("Kate"))[source(self)].
- Rules	loan([Car,Person]) :- purchase(car(Car),driver(Person)).
- Annotations	

Rules: Predicate and Variable

rule | Jason

```
1  day :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3  sky(blue).
4  time(14).
```

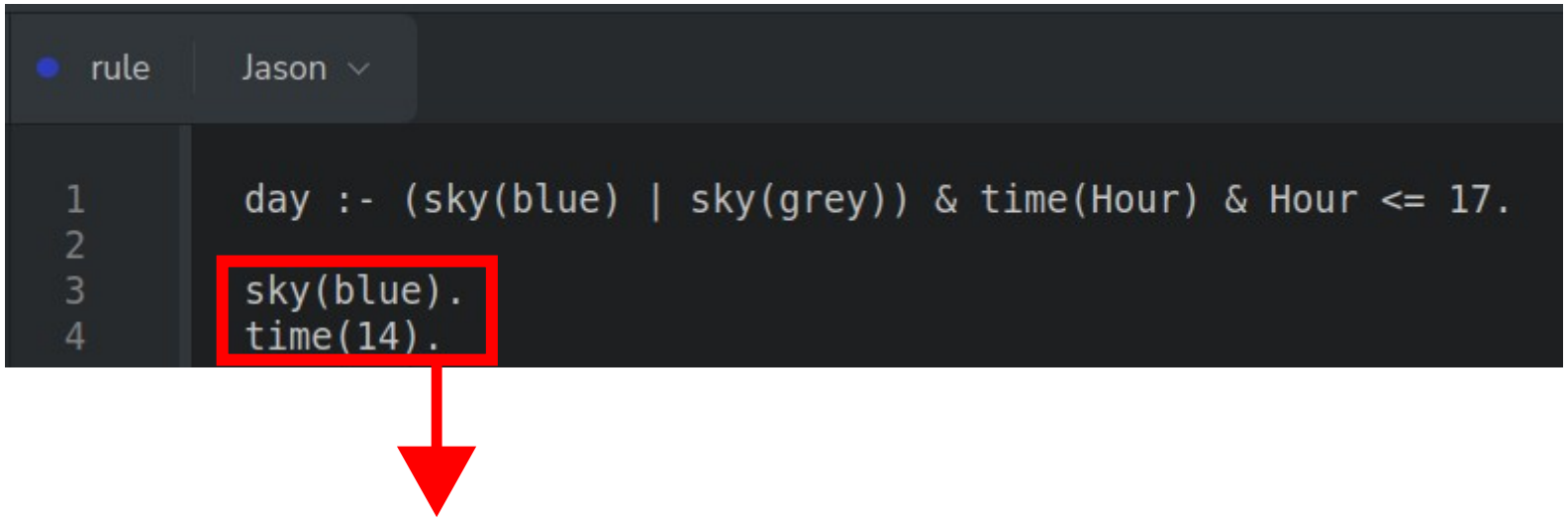
Rules: Predicate and Variable

rule | Jason

```
1   day :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3   sky(blue).
4   time(14).
```

The verified conditions.

Rules: Predicate and Variable



If the beliefs exist...

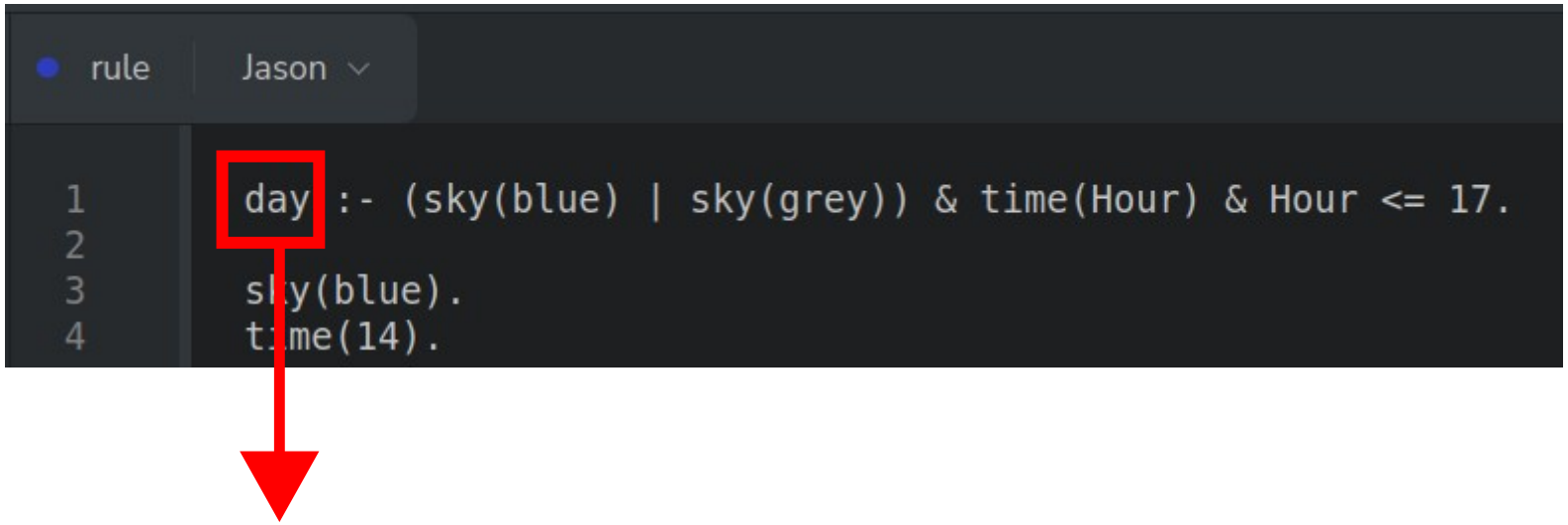
Rules: Predicate and Variable

rule | Jason

```
1   day :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3   sky(blue).
4   time(14).
```

... the predicate holds, and agents can use it ...

Rules: Predicate and Variable



... but it is NOT A BELIEF.

Rules: Predicate and Variable

rule Jason

```
1  day :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3  sky(blue).
4  time(14).
```

Rules: Predicate and Variable

rule | Jason

```
1  day :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3  sky(blue).
4  time(14).
```

Inspection of agent **rule**

- Beliefs	sky(blue)[source(self)]. time(*14*)[source(self)].
- Rules	day :- ((sky(blue) | sky(grey)) & (time(Hour) & (Hour <= *17*))).
- Annotations	

Rules: Underscore (Don’t Care)

rule Jason

```
1  married :- myWife(_).
2
3  myWife(kate).
```

Rules: Underscore (Don’t Care)

Don’t care its value.

Rules: Underscore (Don’t Care)

rule Jason

```
1  married :- myWife(_).
2
3  myWife(kate).
```

Inspection of agent **rule**

- Beliefs	myWife(kate)[source(self)].
- Rules	married :- myWife(_39).
- Annotations	

Rules: Strong Negation

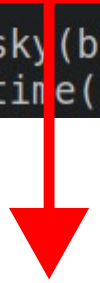
rule | Jason

```
1  ~night :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3  sky(blue).
4  time(14).
```

Rules: Strong Negation

rule Jason

```
1  ~night :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3  sky(blue).
4  time(14).
```



~predicate

Rules: Strong Negation

rule | Jason

```
1   ~night :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3   sky(blue).
4   time(14).
```

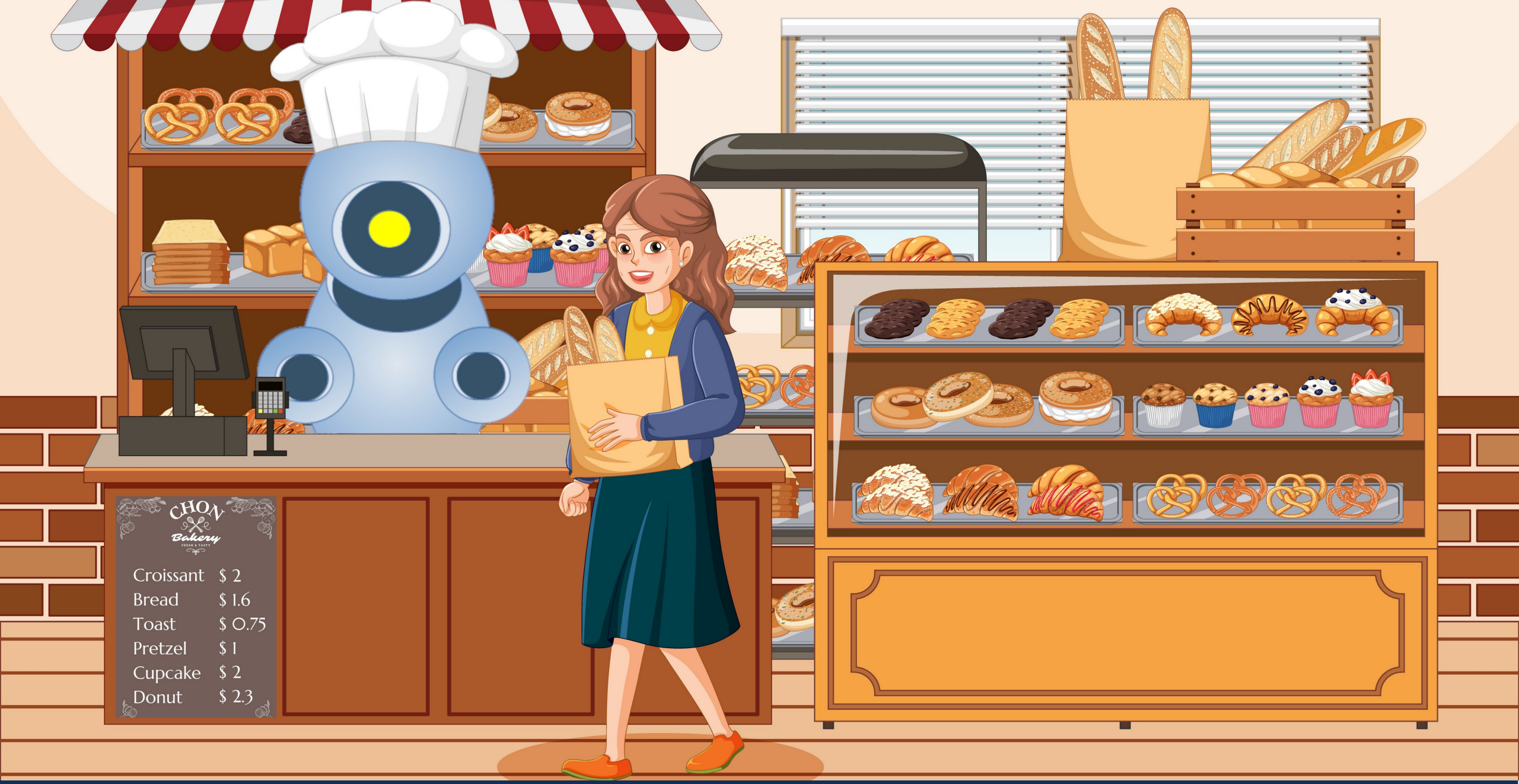
Rules: Strong Negation

rule | Jason

```
1   ~night :- (sky(blue) | sky(grey)) & time(Hour) & Hour <= 17.
2
3   sky(blue).
4   time(14).
```

Inspection of agent **rule**

- Beliefs	sky(blue)[source(self)]. time(*14*)[source(self)].
- Rules	~night :- ((sky(blue) | sky(grey)) & (time(Hour) & (Hour <= *17*))).
- Annotations	


CHON
Bakery
FRESH & TASTY
Croissant $ 2
Bread $ 1.6
Toast $ 0.75
Pretzel $ 1
Cupcake $ 2
Donut $ 2.3

Agradecimentos

OBRIGADO!

pantoja@cefet-rj.br
nilson.lazarin@cefet-rj.br