Sabre
A Narrative Planner Supporting Intention and Deep Theory of Mind

Stephen G. Ware

Cory Siler

Narrative Intelligence LAB

University of Kentucky

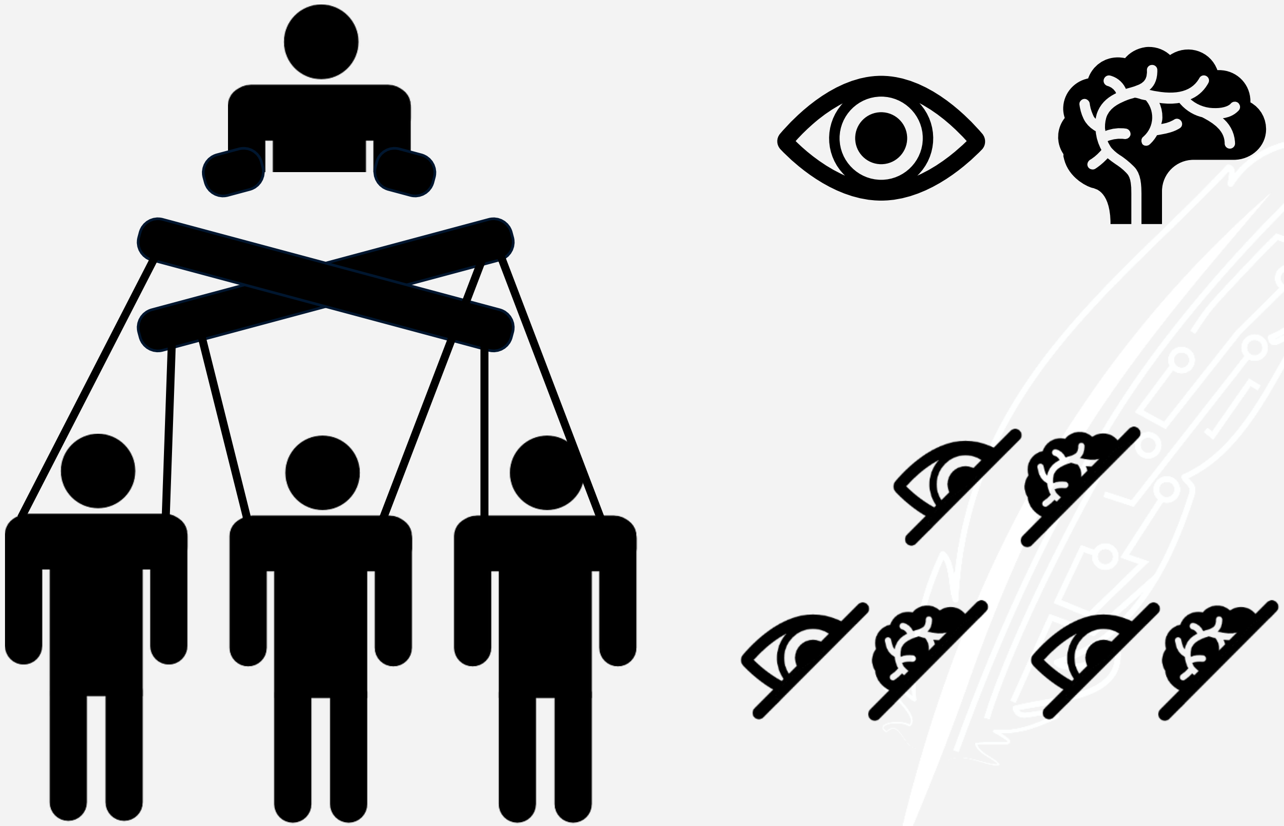
Glaive
NARRATIVE PLANNER

Sabre
NARRATIVE PLANNER

Narrative Planning

A single decision maker

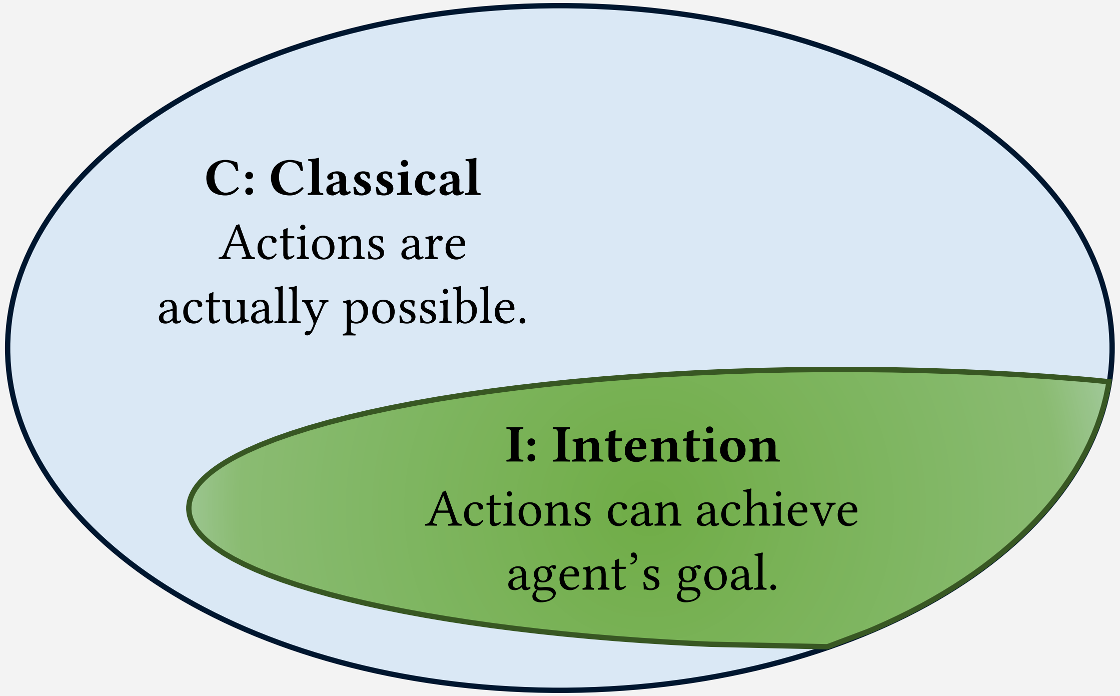
creates the appearance of a multi-agent system.



Intentions and Beliefs

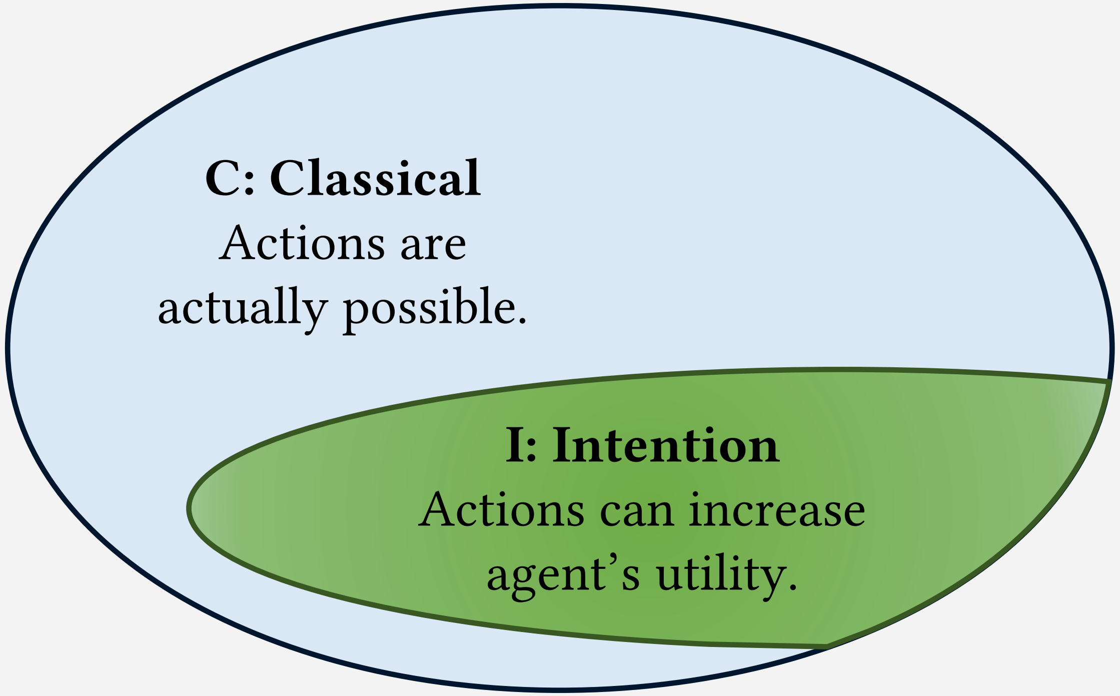
C: Classical
Actions are
actually possible.

Intentions and Beliefs

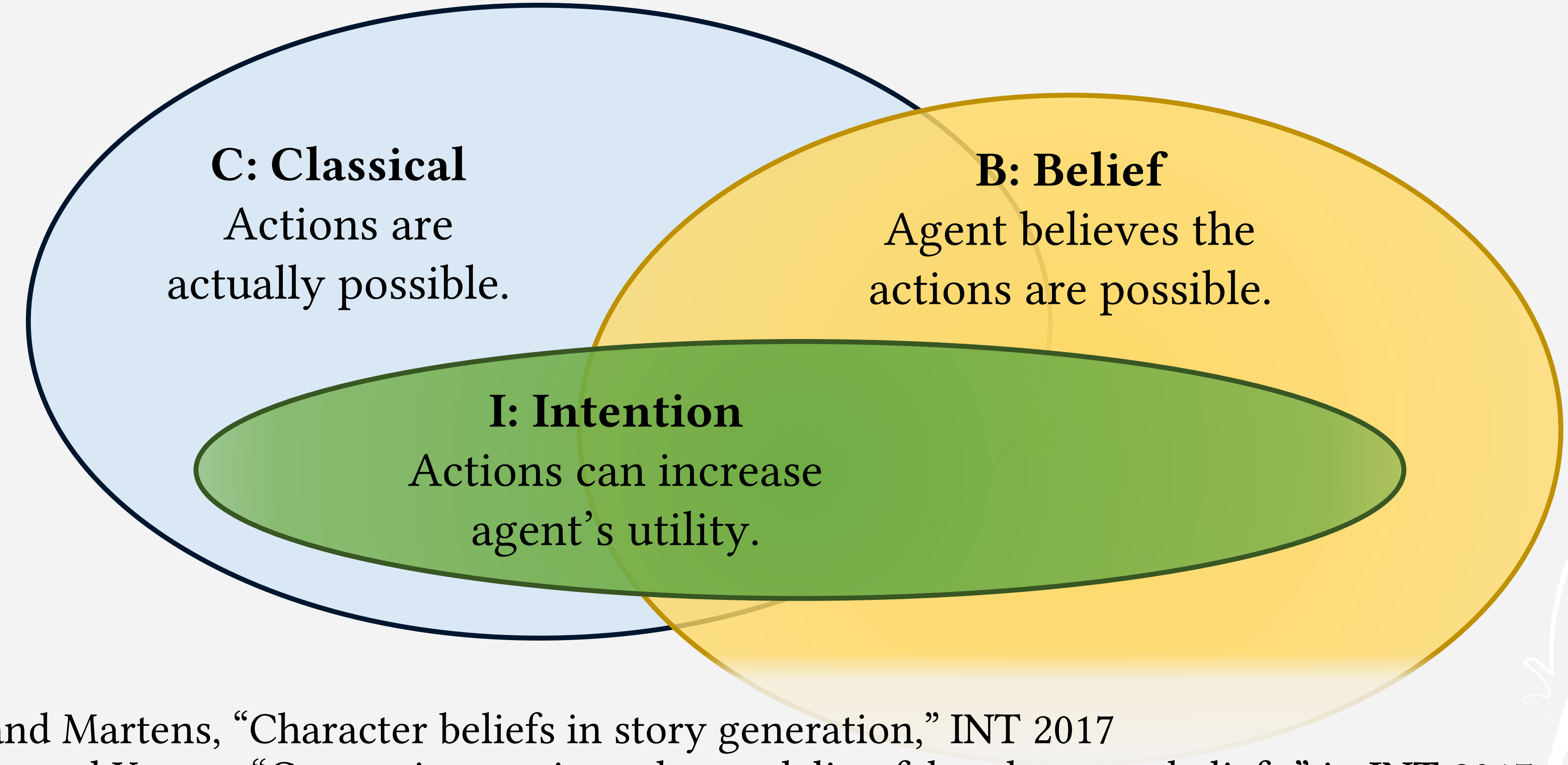


- Riedl and Young, "Narrative planning: balancing plot and character," in JAIR 2010
- Teutenberg and Porteous, "Efficient intent-based narrative generation...," in AAMAS 2013
- Ware and Young, "Glaive: a state-space narrative planner...," in AIIDE 2014

Intentions and Beliefs

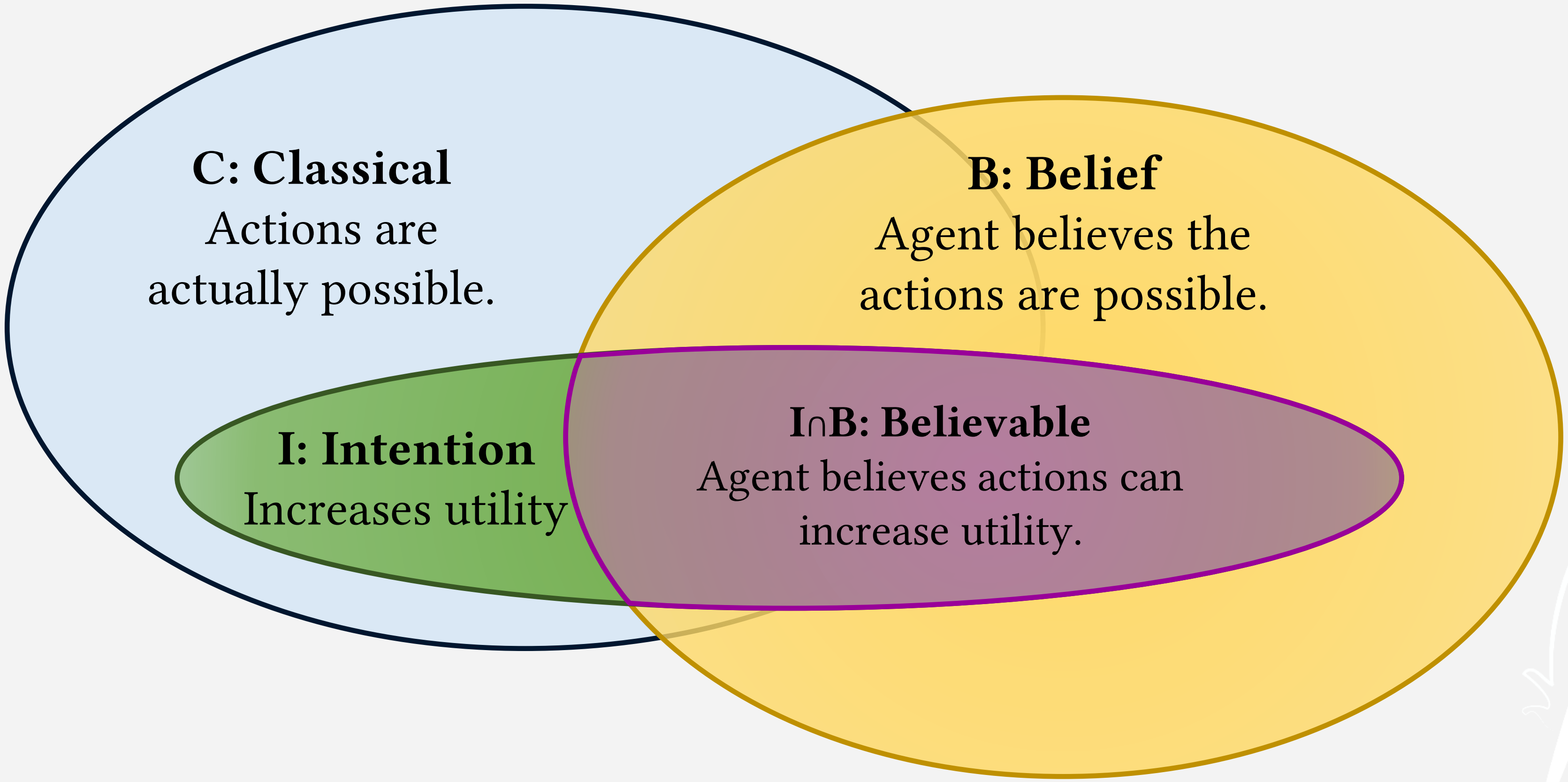


Intentions and Beliefs



- Eger and Martens, "Character beliefs in story generation," INT 2017
- Thorne and Young, "Generating stories … by modeling false character beliefs," in INT 2017
- Shirvani, Ware, and Farrell, "A possible worlds model of belief…," in AIIDE 2017

Intentions and Beliefs



- Shirvani, Farrell, and Ware, "Combining intentionality and belief...," in AIIDE 2018

Syntax and Features

Fluents

$$at(Tom) =$$

Helmert, “The Fast Downward planning system,” in JAIR 2006

Fluents

$$at(Tom) = Cottage$$

Helmert, "The Fast Downward planning system," in JAIR 2006

Fluents

$$at(Tom) = Cottage$$

$$path(Cottage, Market) = \top$$

Fluents

$$at(Tom) = Cottage$$

$$path(Cottage, Market) = \top$$

$$wealth(Merchant) = 3$$

Fluents

$$at(Tom) = Cottage$$

$$path(Cottage, Market) = \top$$

$$wealth(Merchant) = 3$$

$$believes(Tom, wealth(Merchant)) = 2$$

Fluents

$$at(Tom) = Cottage$$

$$path(Cottage, Market) = \top$$

$$wealth(Merchant) = 3$$

$$believes(Tom, wealth(Merchant)) = 2$$

$$believes(Merchant, believes(Tom, wealth(Merchant))) = 3$$

Theory of Mind

- Arbitrarily deep

  what x believes y believes z believes...

- No uncertainty

  Everyone commits to beliefs, which can be wrong.

Other Syntactical Features

- Negation
- Disjunction
- Conditional Effects
- First Order Quantifiers

Actions

$$\boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

Actions

$$a\colon\ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

Actions

$$a:\ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a):$$

Actions

$$\boldsymbol{a}\text{:}\ \ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ \ at(Tom) = Market$$

Actions

$$\boldsymbol{a}\text{:}\ \ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ \ at(Tom) = Market \land at(Merchant) = Market$$

Actions

$$\boldsymbol{a}\text{:}\ \ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ \ at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant$$

NIL

Actions

$$\boldsymbol{a}\text{:}\ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

Actions

$$\boldsymbol{a}\text{:}\quad \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\quad at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a)\text{:}$$

Actions

$$\boldsymbol{a}:\ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a):\ at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a):\ at(Potion) = Tom$$

Actions

$$\boldsymbol{a}\text{:}\ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a)\text{:}\ at(Potion) = Tom \wedge wealth(Merchant) \mathrel{+}= 1$$

Actions

$\boldsymbol{a}$: $\boldsymbol{buy(Tom, Potion, Merchant, Market)}$

PRE(a): $at(Tom) = Market \land at(Merchant) = Market \land$
$at(Potion) = Merchant \land wealth(Tom) \geq 1$

EFF(a): $at(Potion) = Tom \land wealth(Merchant) \mathrel{+}= 1 \land$
$wealth(Tom) \mathrel{-}= 1$

Actions

$$\boldsymbol{a}\text{:}\ \ \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{:}\ \ at(Tom) = Market \wedge at(Merchant) = Market \wedge$$
$$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a)\text{:}\ \ at(Potion) = Tom \wedge wealth(Merchant) \mathrel{+}= 1 \wedge$$
$$wealth(Tom) \mathrel{-}= 1$$

$$\text{CON}(a)\text{:}$$

Actions

$$\boldsymbol{a}\text{: } \boldsymbol{buy(Tom, Potion, Merchant, Market)}$$

$$\text{PRE}(a)\text{: } at(Tom) = Market \wedge at(Merchant) = Market \wedge at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a)\text{: } at(Potion) = Tom \wedge wealth(Merchant) \mathrel{+}= 1 \wedge wealth(Tom) \mathrel{-}= 1$$

$$\text{CON}(a)\text{: } \{Tom, Merchant\}$$

Actions

$\boldsymbol{a}$: $\boldsymbol{buy(Tom, Potion, Merchant, Market)}$

PRE(a): $at(Tom) = Market \wedge at(Merchant) = Market \wedge$
$at(Potion) = Merchant \wedge wealth(Tom) \geq 1$

EFF(a): $at(Potion) = Tom \wedge wealth(Merchant) += 1 \wedge$
$wealth(Tom) -= 1$

CON(a): $\{Tom, Merchant\}$

OBS(a, c):

Actions

$$\boldsymbol{a}\text{: } \boldsymbol{buy}(\boldsymbol{Tom}, \boldsymbol{Potion}, \boldsymbol{Merchant}, \boldsymbol{Market})$$

$$\text{PRE}(a)\text{: } at(Tom) = Market \wedge at(Merchant) = Market \wedge at(Potion) = Merchant \wedge wealth(Tom) \geq 1$$

$$\text{EFF}(a)\text{: } at(Potion) = Tom \wedge wealth(Merchant) \mathrel{+}= 1 \wedge wealth(Tom) \mathrel{-}= 1$$

$$\text{CON}(a)\text{: } \{Tom, Merchant\}$$

$$\text{OBS}(a, c)\text{: } at(c) = Market$$

NIL

Triggers

$$\boldsymbol{t}:\ \boldsymbol{see(Tom, Merchant, Market)}$$

PRE(t):

EFF(t):

Triggers

$\boldsymbol{t}$: $\boldsymbol{see(Tom, Merchant, Market)}$

PRE(t): $at(Tom) = Market$

EFF(t):

Triggers

$$\boldsymbol{t}\text{: } \boldsymbol{see(Tom, Merchant, Market)}$$

$$\text{PRE}(t)\text{: } at(Tom) = Market \wedge at(Merchant) = Market$$

$$\text{EFF}(t)\text{:}$$

Triggers

$\boldsymbol{t}$: $\boldsymbol{see(Tom, Merchant, Market)}$

PRE(t): $at(Tom) = Market \wedge at(Merchant) = Market \wedge$
$believes(Tom, at(Merchant)) \neq Market$

EFF(t):

Triggers

$$\boldsymbol{t}: \ \boldsymbol{see(Tom, Merchant, Market)}$$

$$\text{PRE}(t): \ at(Tom) = Market \land at(Merchant) = Market \land believes(Tom, at(Merchant)) \neq Market$$

$$\text{EFF}(t): \ believes\big(Tom, at(Merchant)\big) = Market$$

Pre-Processing

- Make action and trigger results explicit
- Detect and remove immutable fluents
- Detect and remove impossible actions and triggers

Results of an Event

After Tom buys the potion from the merchant…

- Tom has the potion.
- Tom knows he has the potion.
- The merchant knows Tom has the potion.
- Tom know that the merchant knows that he has the potion.
- … and so on.

Example Trigger: Two-Way Paths

$\boldsymbol{t}$: $\boldsymbol{add\_path}(\boldsymbol{y}, \boldsymbol{x})$

PRE(t) $path(x, y) = \top \wedge path(y, x) = \bot$

EFF(t): $path(y, x) = \top$

Example Trigger: Two-Way Paths

$$\boldsymbol{t}\text{: } \boldsymbol{add\_path(Market, Cottage)}$$

$$\text{PRE}(t)\text{: } path(Cottage, Market) = \top \wedge$$

$$path(Market, Cottage) = \bot$$

$$\text{EFF}(t)\text{: } path(Market, Cottage) = \top$$

NIL

Example Action: Walk

$$\boldsymbol{a}: \ \boldsymbol{walk(Tom, Market, Cottage)}$$

$$\text{PRE}(a): \ at(Tom) = Market \wedge$$
$$path(Market, Cottage) = \top$$

$$\text{EFF}(a): \ at(Tom) = Cottage$$

$$\text{CON}(a): \ \{Tom\}$$

$$\text{OBS}(a, c): \ at(c) = Market \vee at(c) = Cottage$$

Example Action: Walk

$$\boldsymbol{a}\text{: } \boldsymbol{walk(Tom, Market, Cottage)}$$

$$\text{PRE}(a)\text{: } at(Tom) = Market \wedge$$

$$path(Market, Cottage) = \top$$

$$\text{EFF}(a)\text{: } at(Tom) = Cottage$$

$$\text{CON}(a)\text{: } \{Tom\}$$

$$\text{OBS}(a, c)\text{: } at(c) = Market \vee at(c) = Cottage$$

Example Action: Walk

$$\boldsymbol{a}:\ \boldsymbol{walk(Tom, Market, Cottage)}$$

$$\text{PRE}(a):\ at(Tom) = Market$$

$$\text{EFF}(a):\ at(Tom) = Cottage$$

$$\text{CON}(a):\ \{Tom\}$$

$$\text{OBS}(a, c):\ at(c) = Market \lor at(c) = Cottage$$

Search

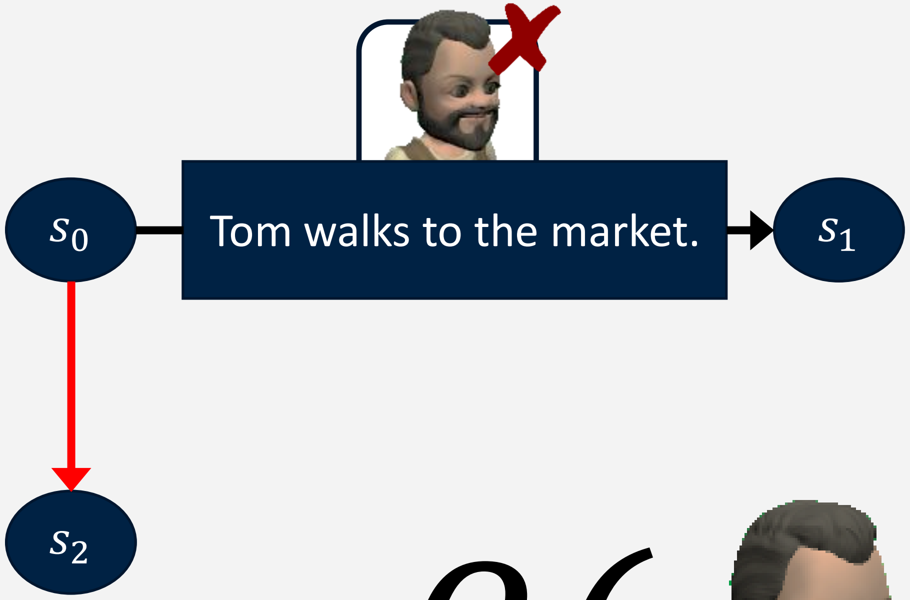
Algorithm 1 The Sabre algorithm

Let $\mathcal{A}$ be the set of all actions defined in the domain.
$\textsc{Sabre}(c_{author}, s_0, \emptyset, s_0)$
function $\textsc{Sabre}(c, r, \pi, s)$
    Input: character c, start state r, plan π, current state s
    if $u(c, s) > u(c, r)$ and π is non-redundant **then**
        return π
    Choose an action $a \in \mathcal{A}$ such that $s \models \textsc{Pre}(a)$.
    for all $c' \in \textsc{Con}(a)$ such that $c' \neq c$ **do**
        Let state $b = \alpha(a, \beta(c', s))$.
        if b is undefined **then return** failure.
        else if $\textsc{Sabre}(c', b, \emptyset, b)$ fails **then return** failure.
    return $\textsc{Sabre}(c, r, \pi \cup a, \alpha(a, s))$

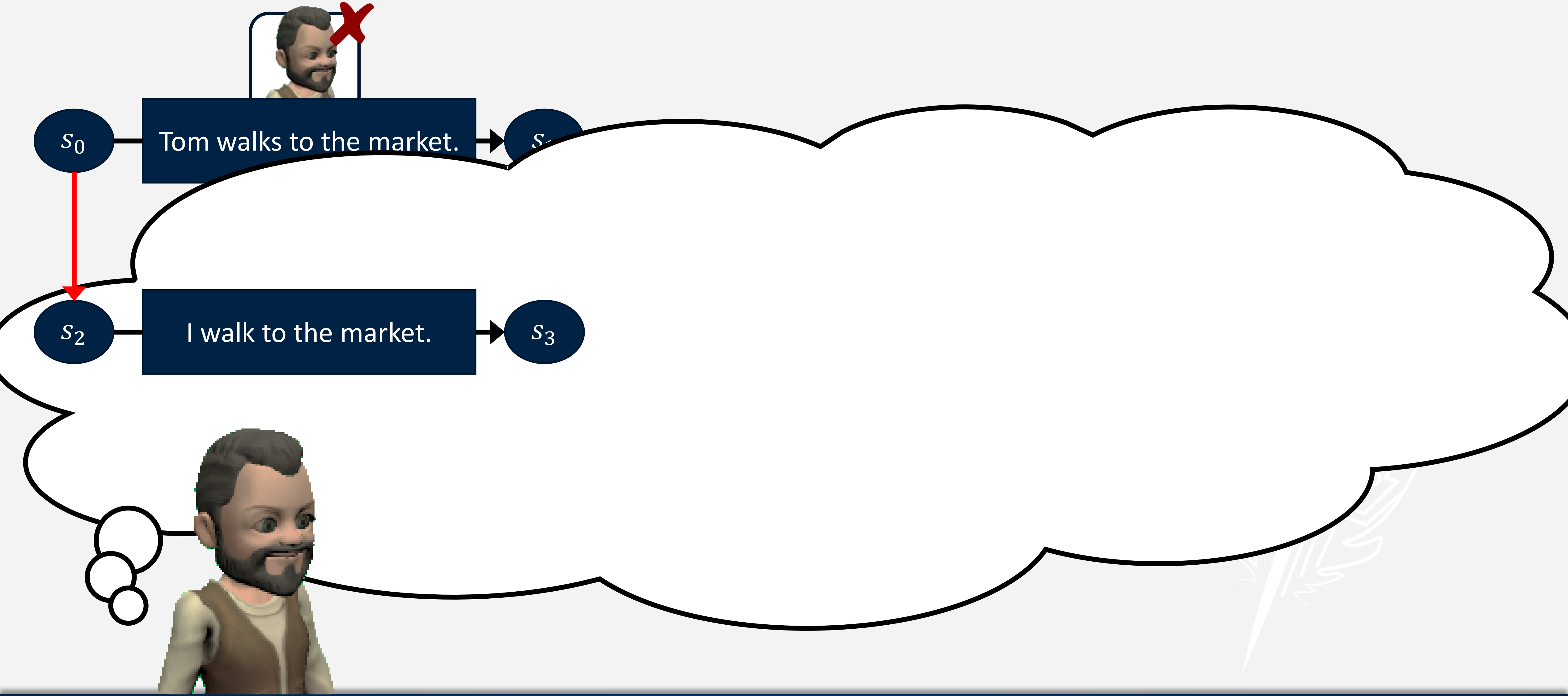
s_0

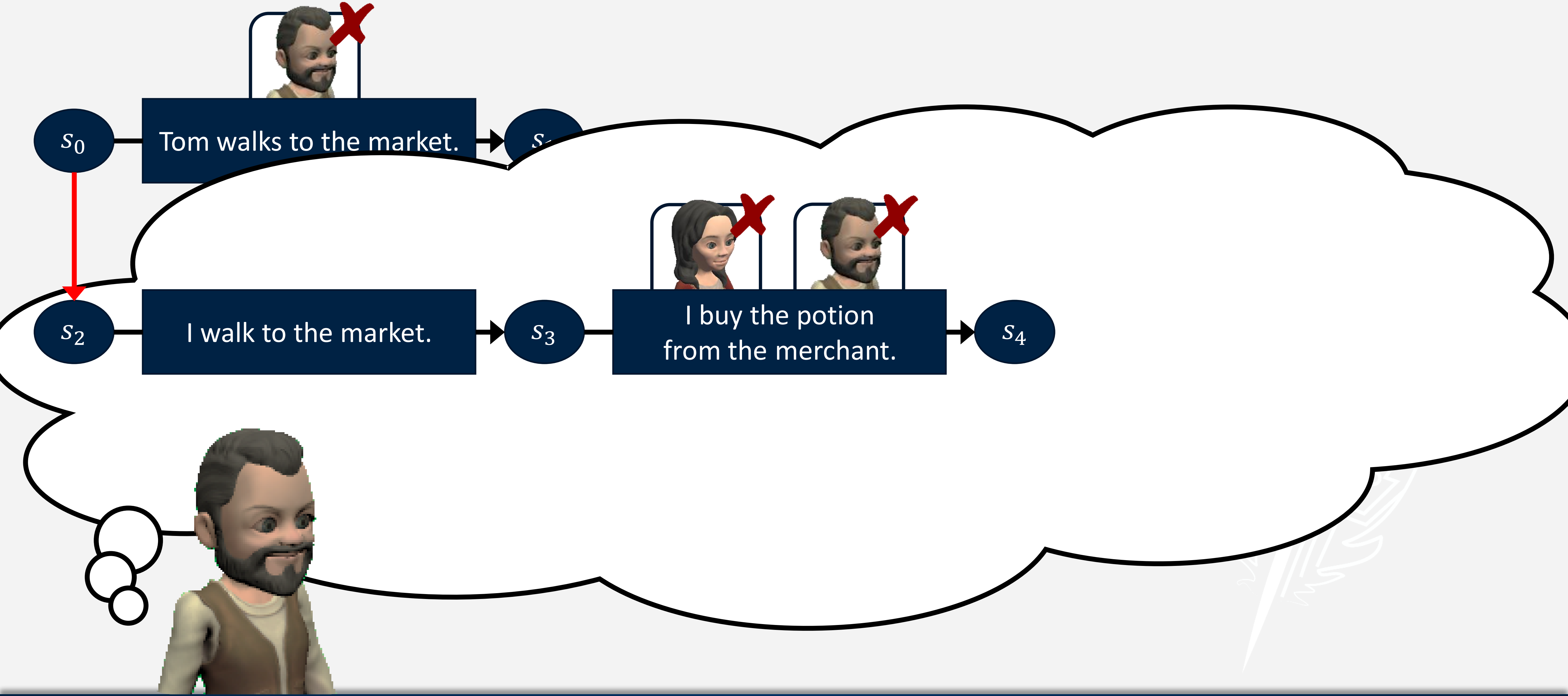
s_0

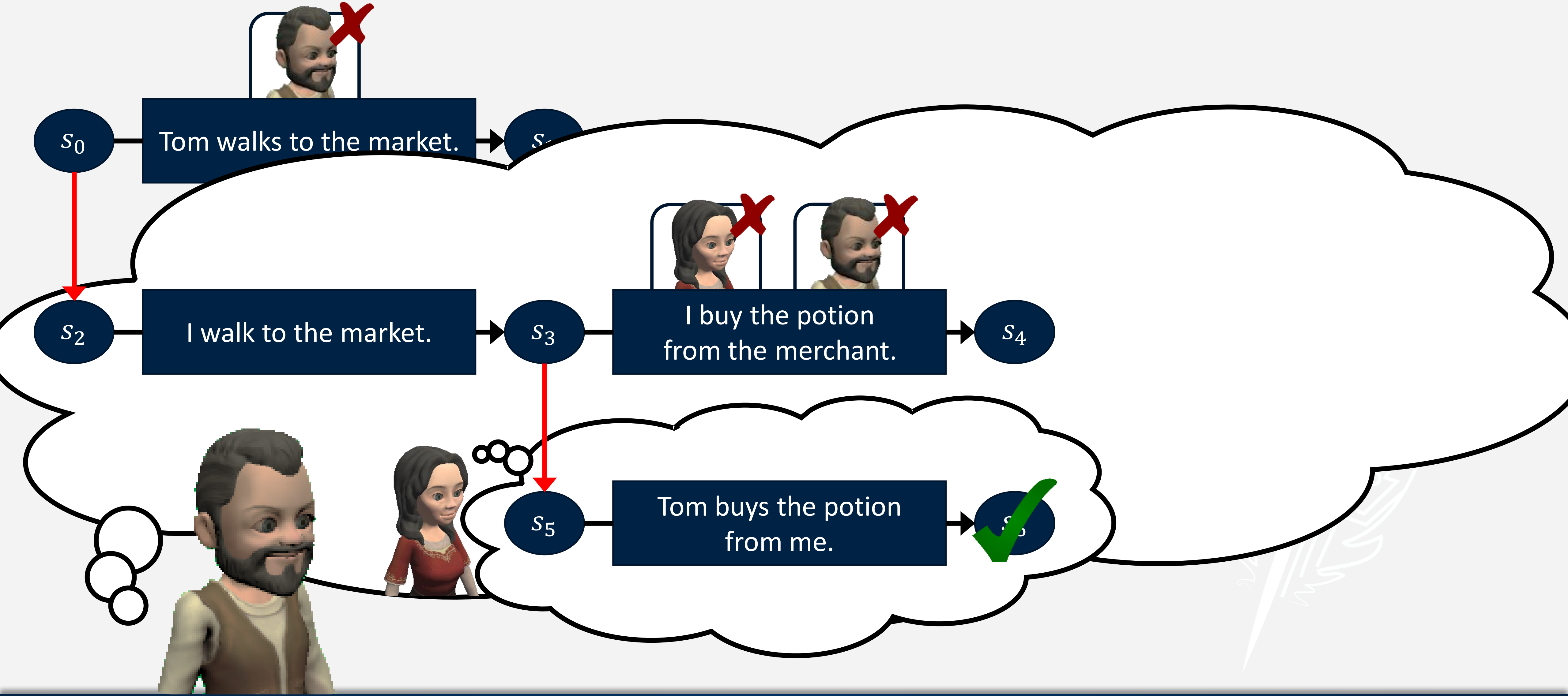
$$\alpha(\text{Tom walks to the market.}, s_0) = s_1$$

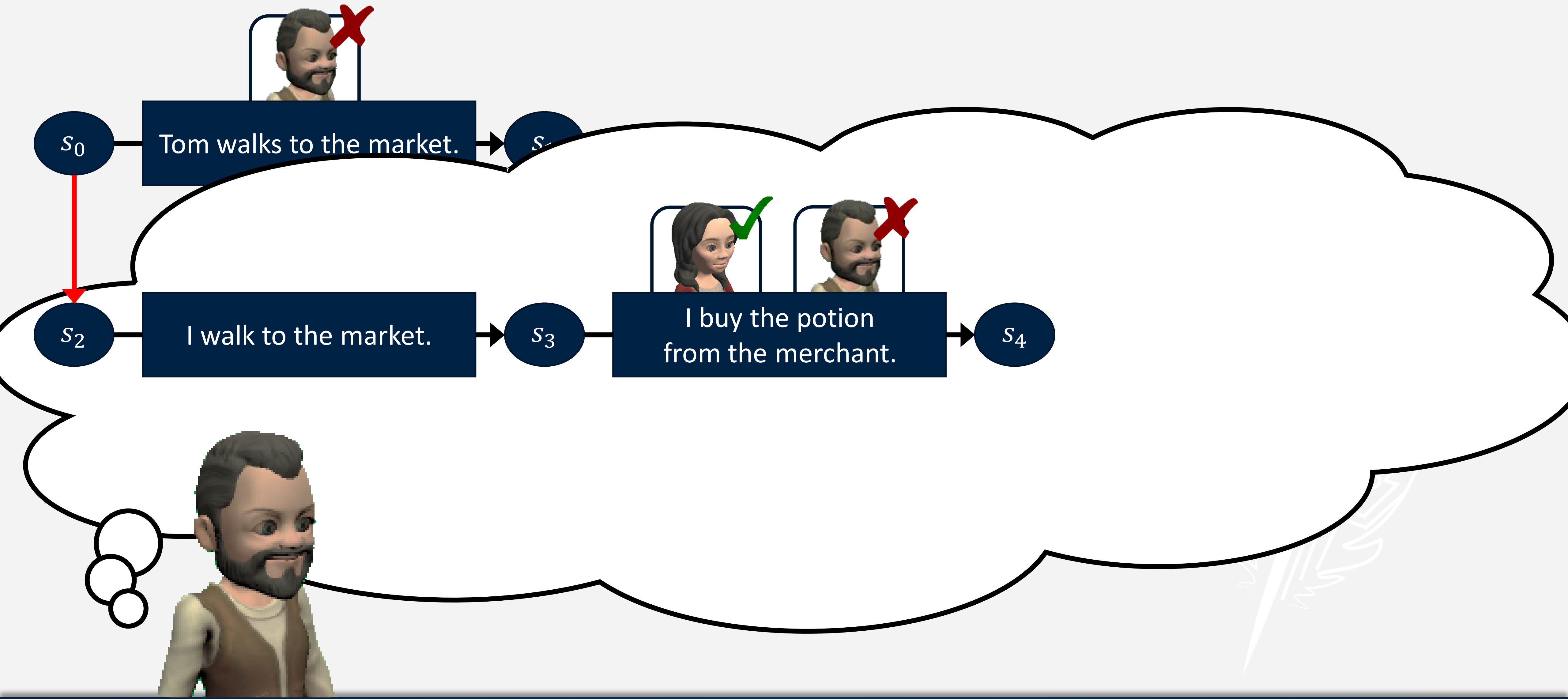


$\beta($  $, s_0) = s_2$

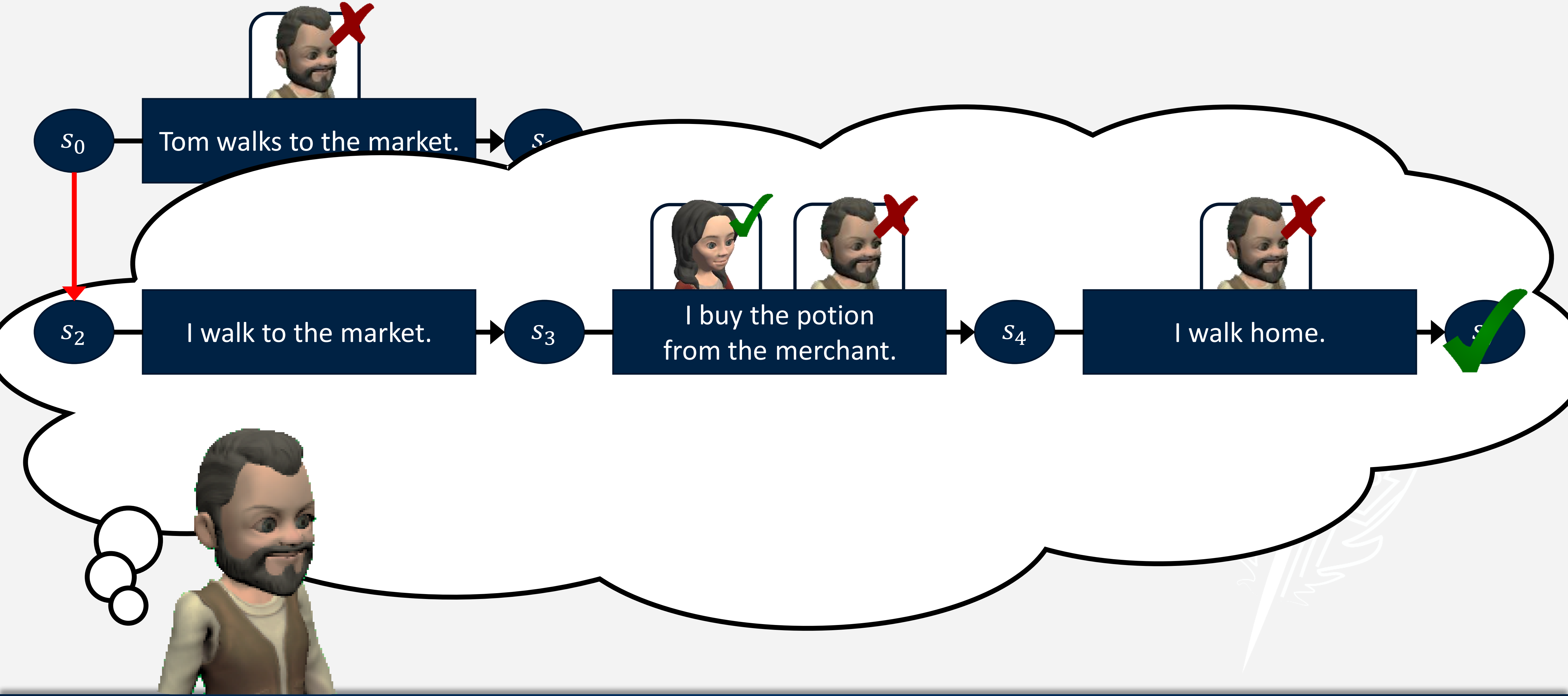

s_0
Tom walks to the market.
s_2
I walk to the market.
s_3

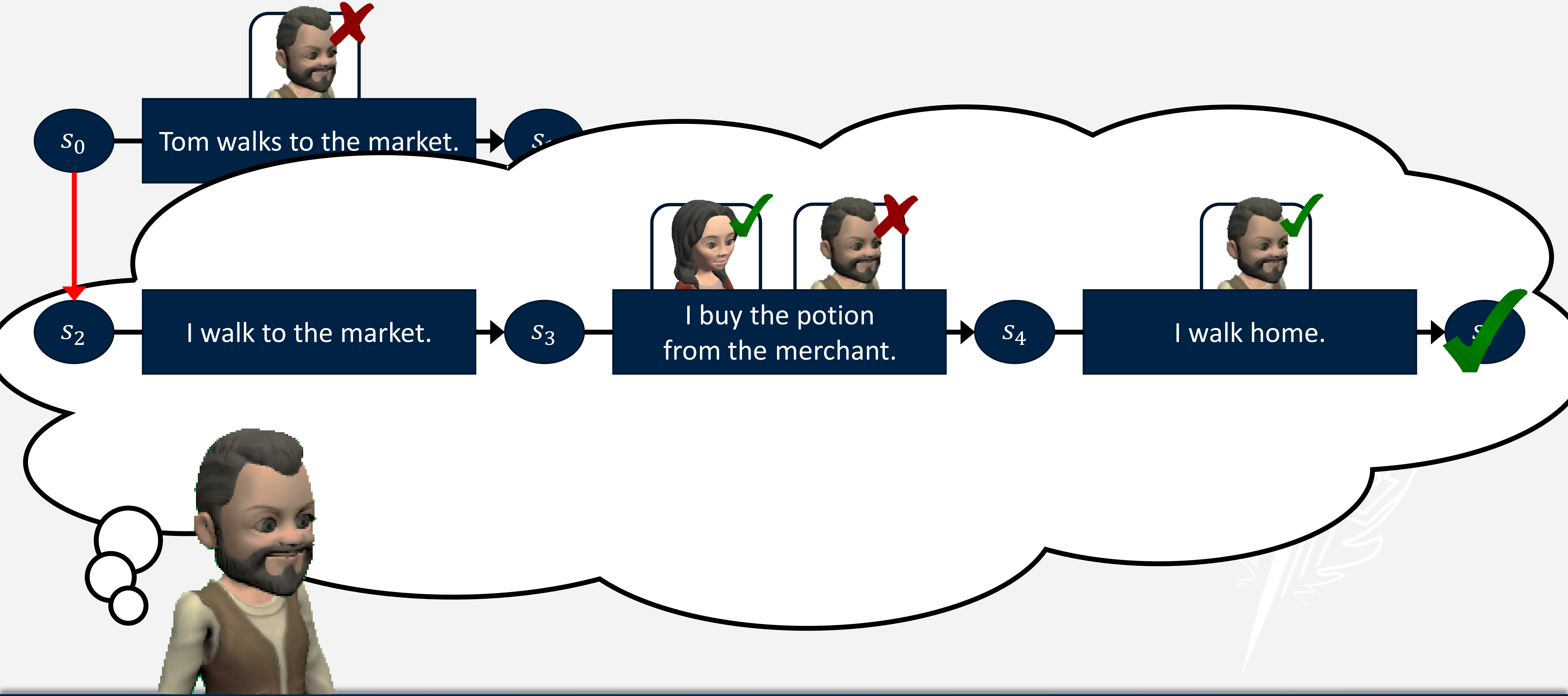

s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion
from the merchant.
s_4


s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion from the merchant.
s_4
s_5
Tom buys the potion from me.

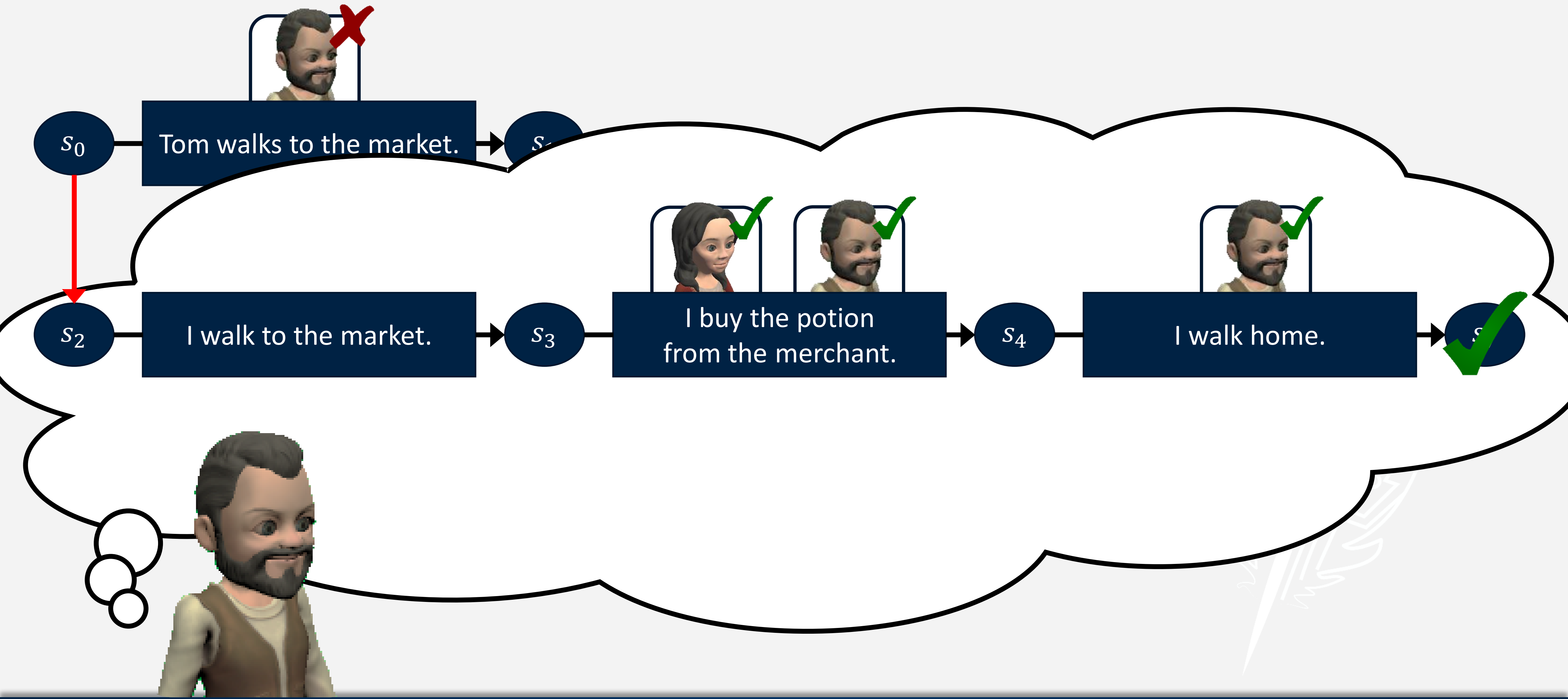


s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion from the merchant.
s_4

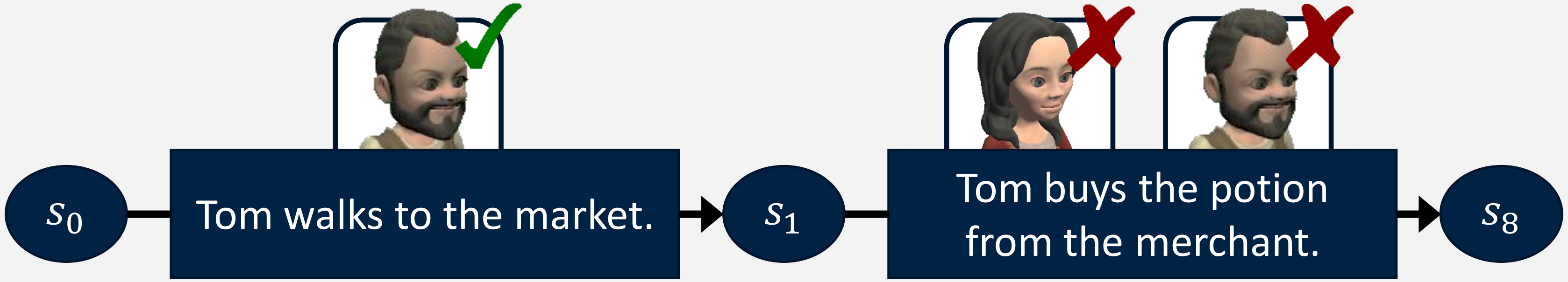

s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion from the merchant.
s_4
I walk home.

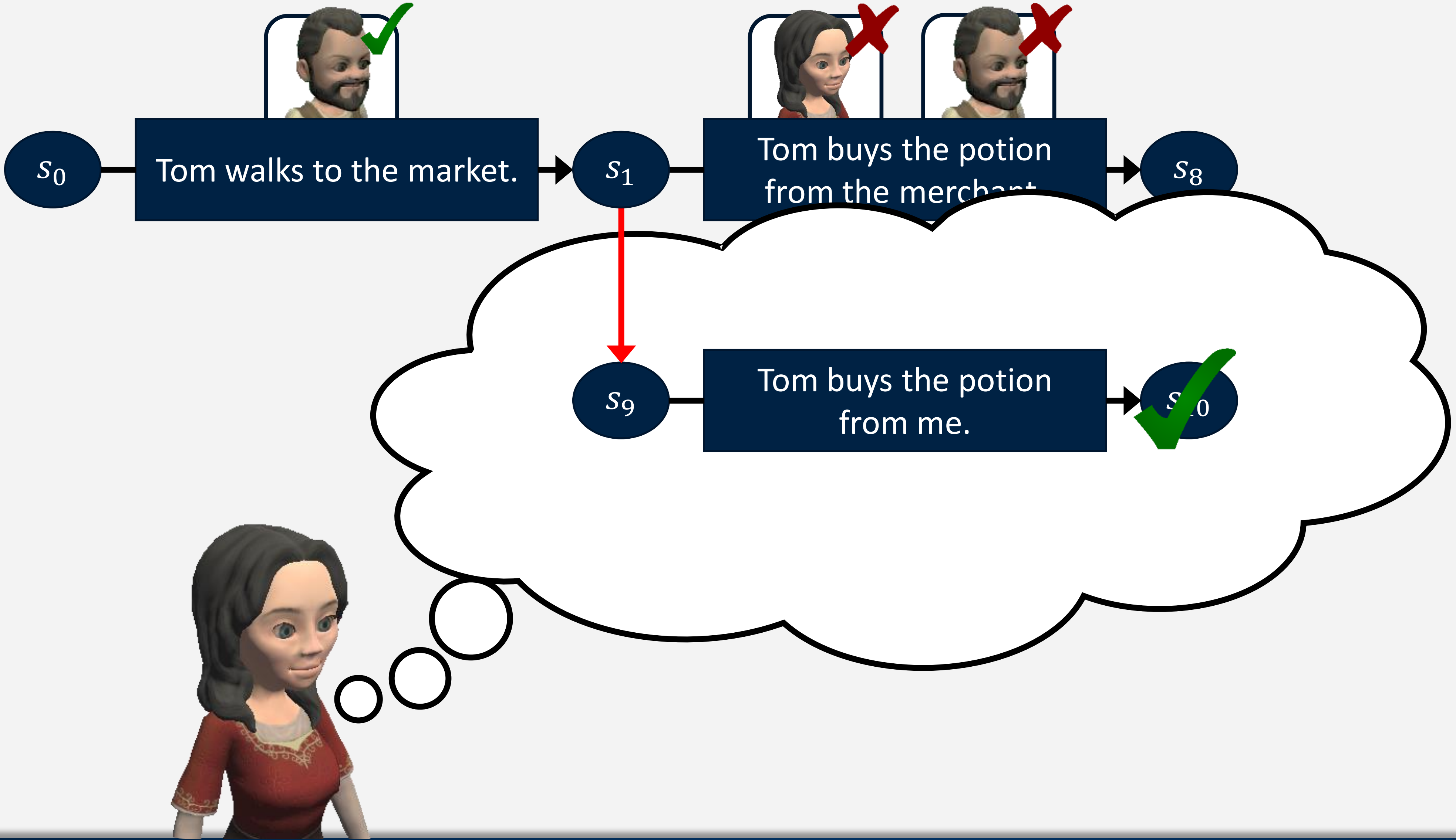


s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion from the merchant.
s_4
I walk home.

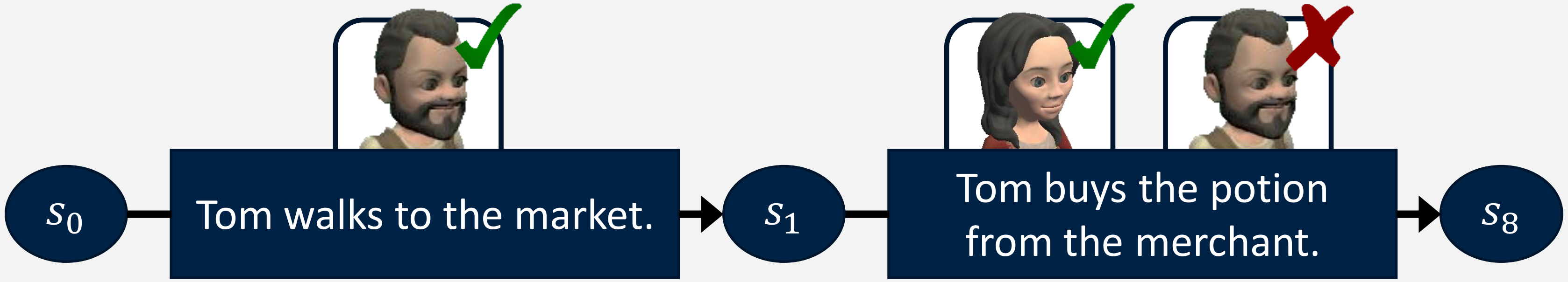


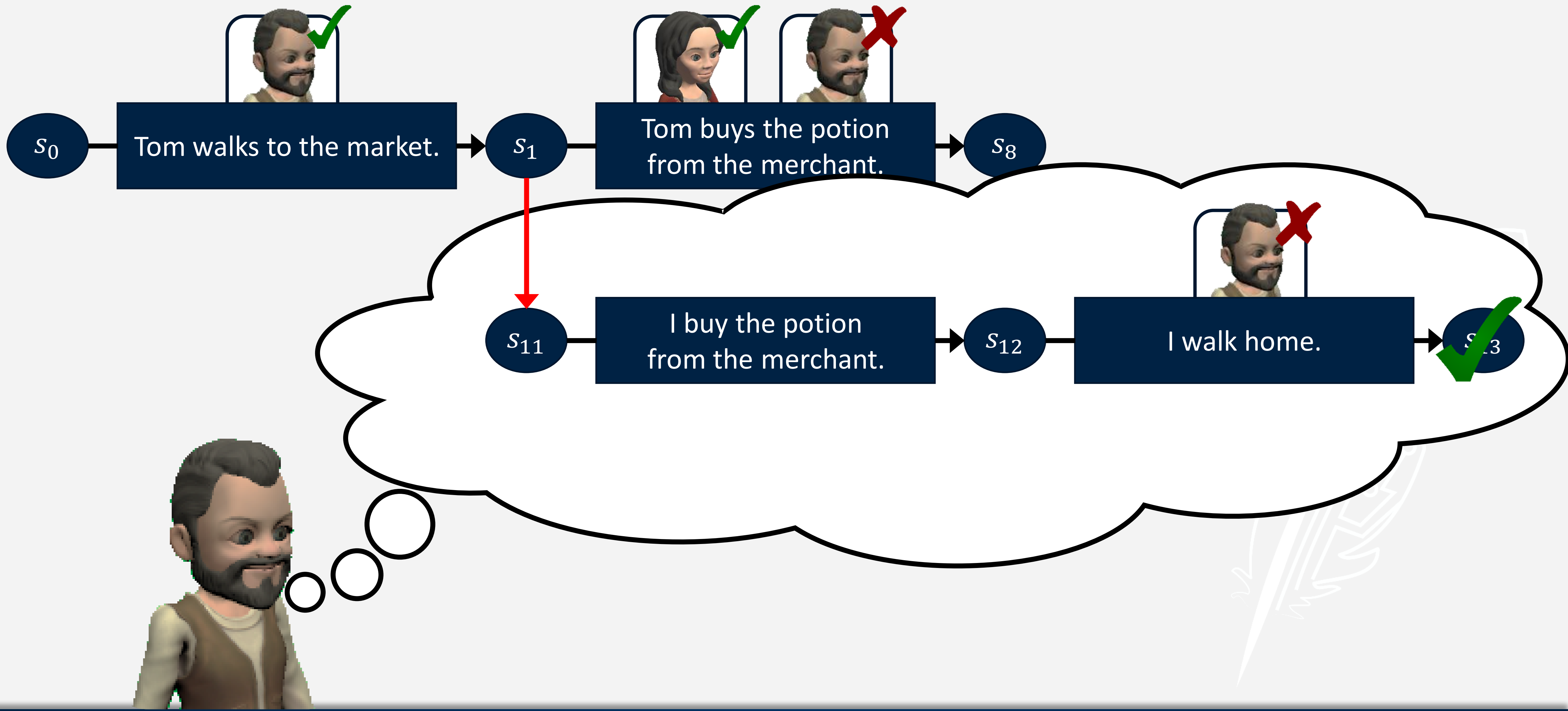
s_0
Tom walks to the market.
s_2
I walk to the market.
s_3
I buy the potion from the merchant.
s_4
I walk home.


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8

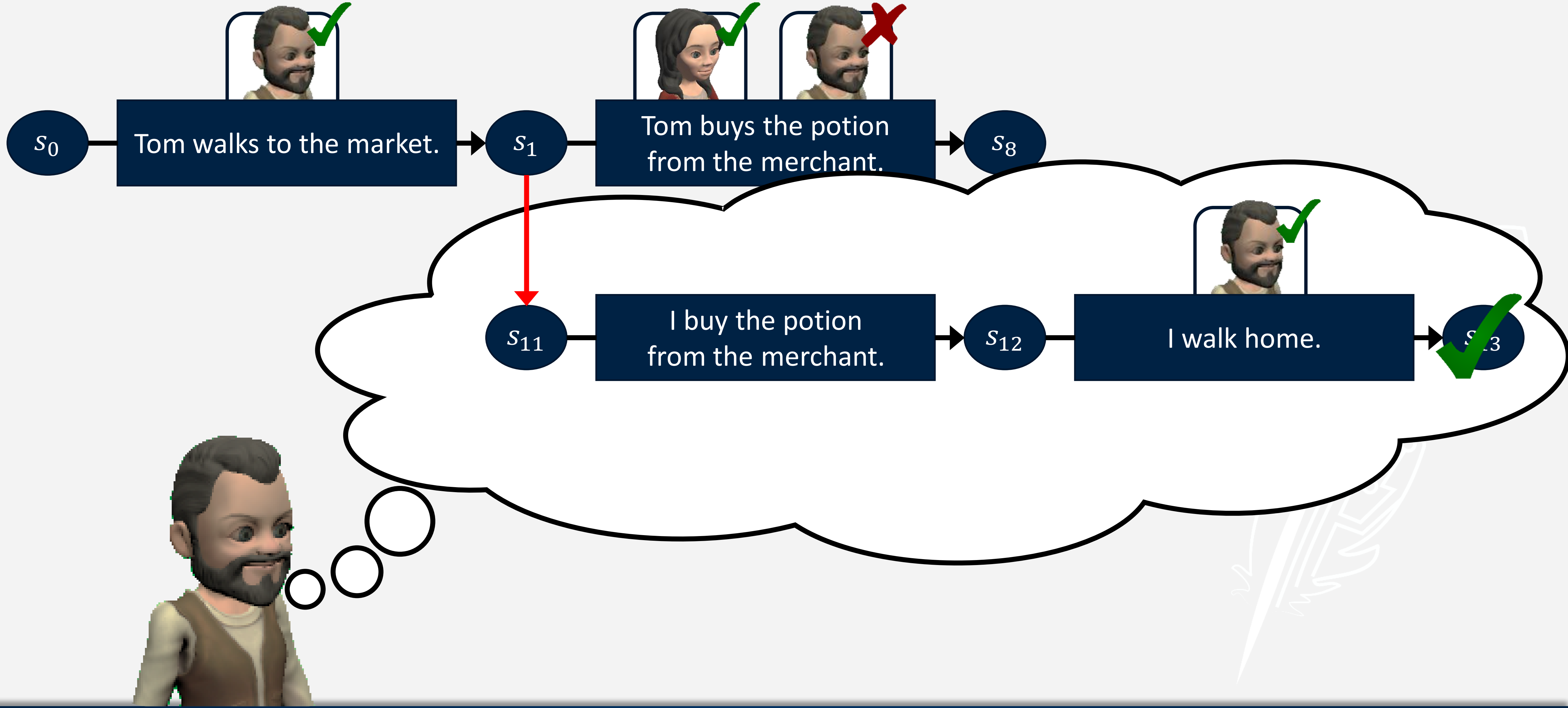


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
s_9
Tom buys the potion from me.
s_{10}

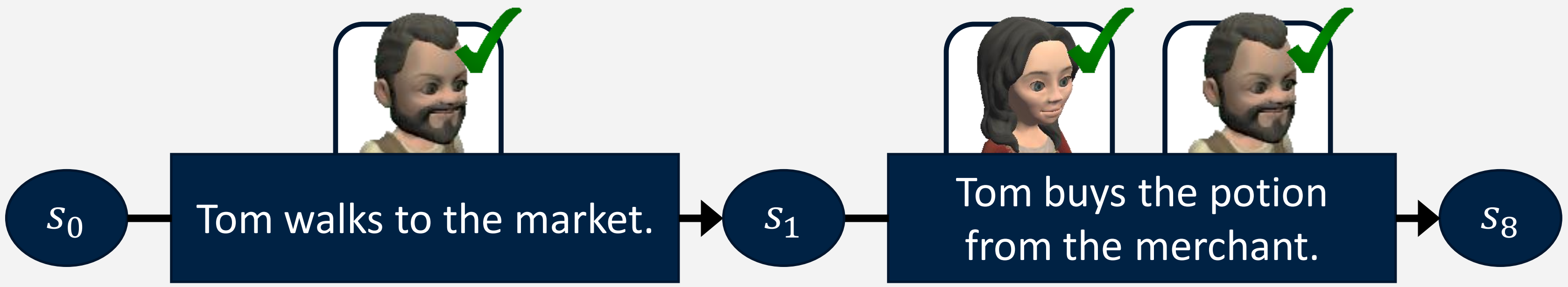

s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8



s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
s_{11}
I buy the potion from the merchant.
s_{12}
I walk home.

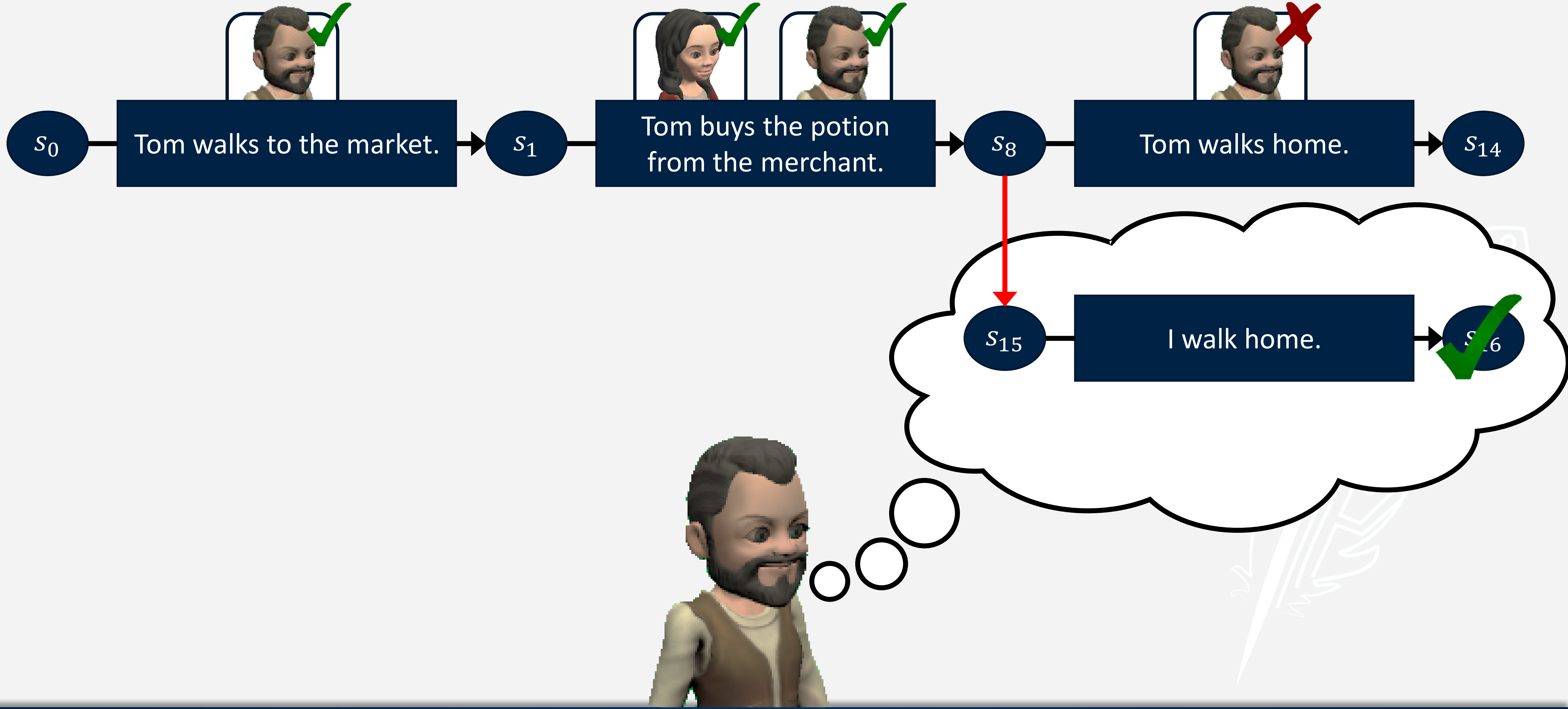


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
s_{11}
I buy the potion from the merchant.
s_{12}
I walk home.



s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
Tom walks home.
s_{14}


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
Tom walks home.
s_{14}
s_{15}
I walk home.
s_{16}


s_0
Tom walks to the market.
s_1
Tom buys the potion from the merchant.
s_8
Tom walks home.
s_{14}

Evaluation

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗
Glaive	✓	✓	✗	✗

- Riedl and Young, “Narrative planning: balancing plot and character,” in JAIR 2010
- Ware and Young, “CPOCL: a narrative planner supporting conflict,” in AIIDE 2011
- Teutenberg and Porteous, “Efficient intent-based narrative generation…,” in AAMAS 2013
- Ware and Young, “Glaive: a state-space narrative planner…,” in AIIDE 2014

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗
Glaive	✓	✓	✗	✗
HeadSpace	✓	✗	~✓	✗

- Thorne and Young, “Generating stories … by modeling false character beliefs,” in INT 2017

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗
Glaive	✓	✓	✗	✗
HeadSpace	✓	✗	~✓	✗
IMPRACTical	✓	✓	~✓	✗

- Teutenberg and Porteous, "Incorporating global and local knowledge…," in AAMAS 2015

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗
Glaive	✓	✓	✗	✗
HeadSpace	✓	✗	~✓	✗
IMPRACTical	✓	✓	~✓	✗
Thespian	✗	✓	✓	✓

- Ryan, Summerville, Mateas, and Wardrip-Fruin, “Toward characters who observe…,” in EXAG 2015
- Si and Marsella, “Encoding Theory of Mind in character design…,” in AHCI 2014

Comparing Sabre to Other Planners

	Centralized	Intentions	Beliefs	Uncertainty
Sabre	✓	✓	✓	✗
Glaive	✓	✓	✗	✗
HeadSpace	✓	✗	~✓	✗
IMPRACTical	✓	✓	~✓	✗
Thespian	✗	✓	✓	✓
Ostari	✓	✓	✓	✓

- Eger and Martens, "Practical specification of belief manipulation in games," in AIIDE 2017

Test Problems

- *Raiders*
- *Space*

- Ware and Young, "Glaive: a state-space narrative planner...," in AIIDE 2014

Test Problems

- *Raiders*
- *Space*
- *Treasure*
- *Lovers*
- *Hubris*

- Farrell and Ware, "Narrative planning for belief and intention recognition," in AIIDE 2020
- Shirvani, Farrell, and Ware, "Combining intentionality and belief ...," in AIIDE 2018
- Christensen, Nelson, and Cardona-Rivera, "Using domain compilation to add belief ...," in AIIDE 2020

Test Problems

- *Raiders*
- *Space*
- *Treasure*
- *Lovers*
- *Hubris*
- *BearBirdJr*

- Sack, "Micro-TaleSpin, a story generator," 1992
- Meehan, "TALE-SPIN, an interactive program that writes stories," in AAAI 1977

Test Problems

- *Raiders*
- *Space*
- *Treasure*
- *Lovers*
- *Hubris*
- *BearBirdJr*
- *Grandma*

- Ware, Garcia, Shirvani, and Farrell, "Multi-agent experience management …," in AIIDE 2019

Results

Domain	Nodes Generated	Time
Raiders	17,815	1.4 s
Space	192	6 ms
Treasure	288	1 ms
Lovers	5,198,414	40.3 m
Hubris	831	47 ms
BearBirdJr	34,084,068	14.0 m
Grandma	105,178,466	6.2 h

Conclusion

Limitations

- No true uncertainty
- h^+ heuristic often performs poorly[1]

1. Bonet and Geffner, "Planning as heuristic search," in AI, 2001

Future Work

- More search methods

Algorithm 1 The Sabre algorithm

Let $\mathcal{A}$ be the set of all actions defined in the domain.
$\textsc{Sabre}(c_{author}, s_0, \emptyset, s_0)$
function $\textsc{Sabre}(c, r, \pi, s)$
    Input: character c, start state r, plan π, current state s
    if $u(c, s) > u(c, r)$ and π is non-redundant **then**
        return π
    Choose an action $a \in \mathcal{A}$ such that $s \models \textsc{pre}(a)$
    for all $c' \in \textsc{con}(a)$ such that $c' \neq c$ **do**
        Let state $b = \alpha(a, \beta(c', s))$.
        if b is undefined **then return** failure.
        else if $\textsc{Sabre}(c', b, \emptyset, b)$ fails **then return** failure.
    return $\textsc{Sabre}(c, r, \pi \cup a, \alpha(a, s))$

NIL

Future Work

- More search methods

Algorithm 1 The Sabre algorithm

Let $\mathcal{A}$ be the set of all actions defined in the domain.
$\textsc{Sabre}(c_{author}, s_0, \emptyset, s_0)$
function $\textsc{Sabre}(c, r, \pi, s)$
  Input: character c, start state r, plan π, current state s
  if $u(c, s) > u(c, r)$ and π is non-redundant **then**
    return π
  Choose an action $a \in \mathcal{A}$ such that $s \models \textsc{pre}(a)$.
  for all $c' \in \textsc{con}(a)$ such that $c' \neq c$ **do**
    Let state $b = \alpha(a, \beta(c', s))$.
    if b is undefined **then return** failure.
    else if $\textsc{Sabre}(c', b, \emptyset, b)$ fails **then return** failure.
  return $\textsc{Sabre}(c, r, \pi \cup a, \alpha(a, s))$

Algorithm 2 The Sabre algorithm

Let $\mathcal{A}$ be the set of all actions defined in the domain.
$\textsc{Sabre}(c_{author}, s_0, \emptyset, s_0)$
function $\textsc{Sabre}(c, r, \pi, s)$
  Input: character c, start state r, plan π, current state s
  if $u(c, s) > u(c, r)$ and π is non-redundant **then**
    return π
  Choose an action $a \in \mathcal{A}$ such that $s \models \textsc{pre}(a)$.
  if $\textsc{Sabre}(c, r, \pi \cup a, \alpha(a, s))$ fails **then return** failure.
  for all $c' \in \textsc{con}(a)$ such that $c' \neq c$ **do**
    Let state $b = \alpha(a, \beta(c', s))$.
    if b is undefined **then return** failure.
    else if $\textsc{Sabre}(c', b, \emptyset, b)$ fails **then return** failure.
  return π

Future Work

- More search methods
- Better heuristics
- Agent emotions and personalities[1]

1. Shirvani and Ware, "A formalization of emotional planning for strong-story systems," in AIIDE 2020

SABRE
NARRATIVE PLANNER

http://cs.uky.edu/~sgware/projects/sabre

Background Music: https://www.bensound.com