Multi Agent Information Gathering from Public Sources

Gerd Beuster
gb@uni-koblenz.de

KI RESEARCH GROUP

University Koblenz-Landau
Germany

Overview

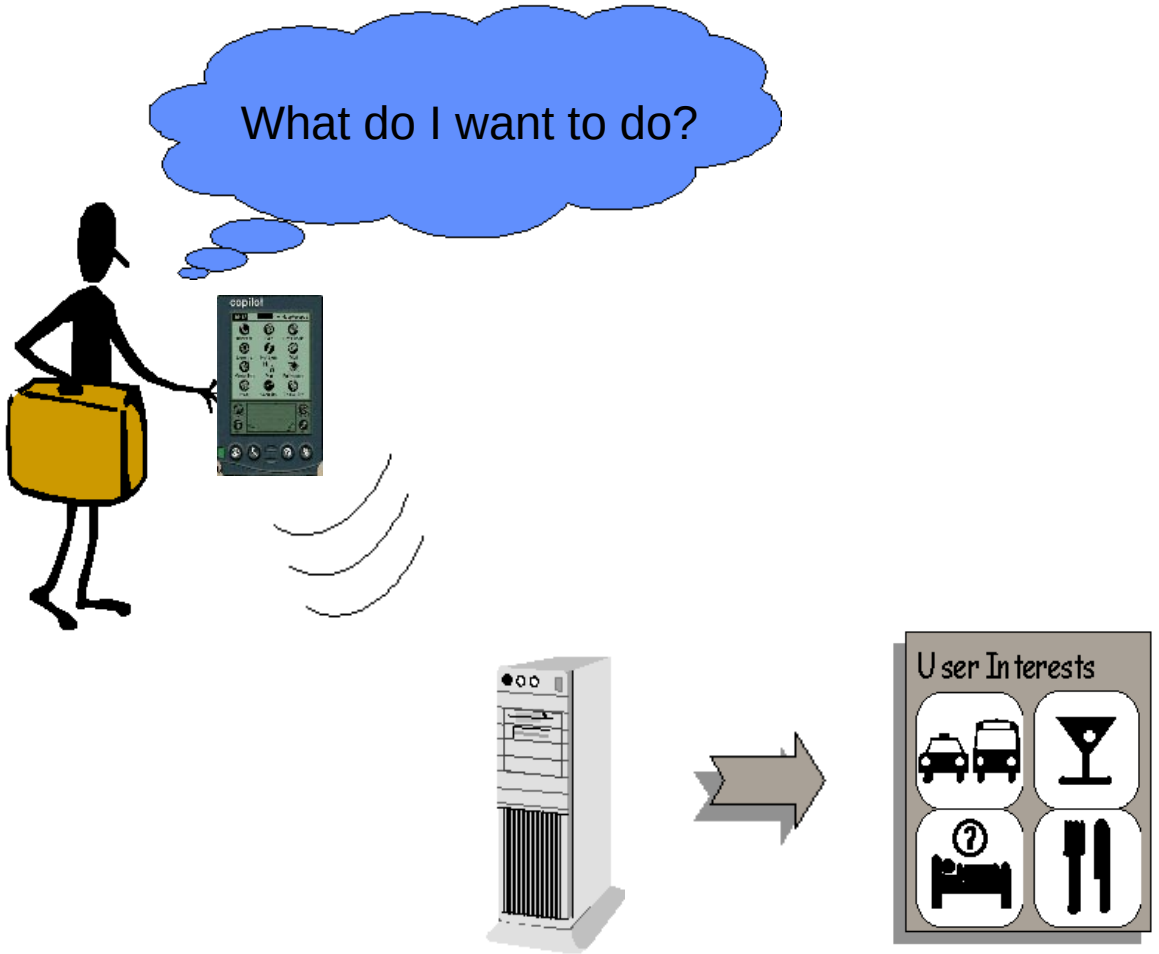
- System MIA
  - MIA — An agent based search engine
  - Accessing heterogenous databases within MIA
  - MIA's logic agents
- Communication among Information Agents
  - Communication layer model
  - Message transportation problems
  - Security
  - Languages and ontologies
- Conclusions

MIA — An agent based search engine

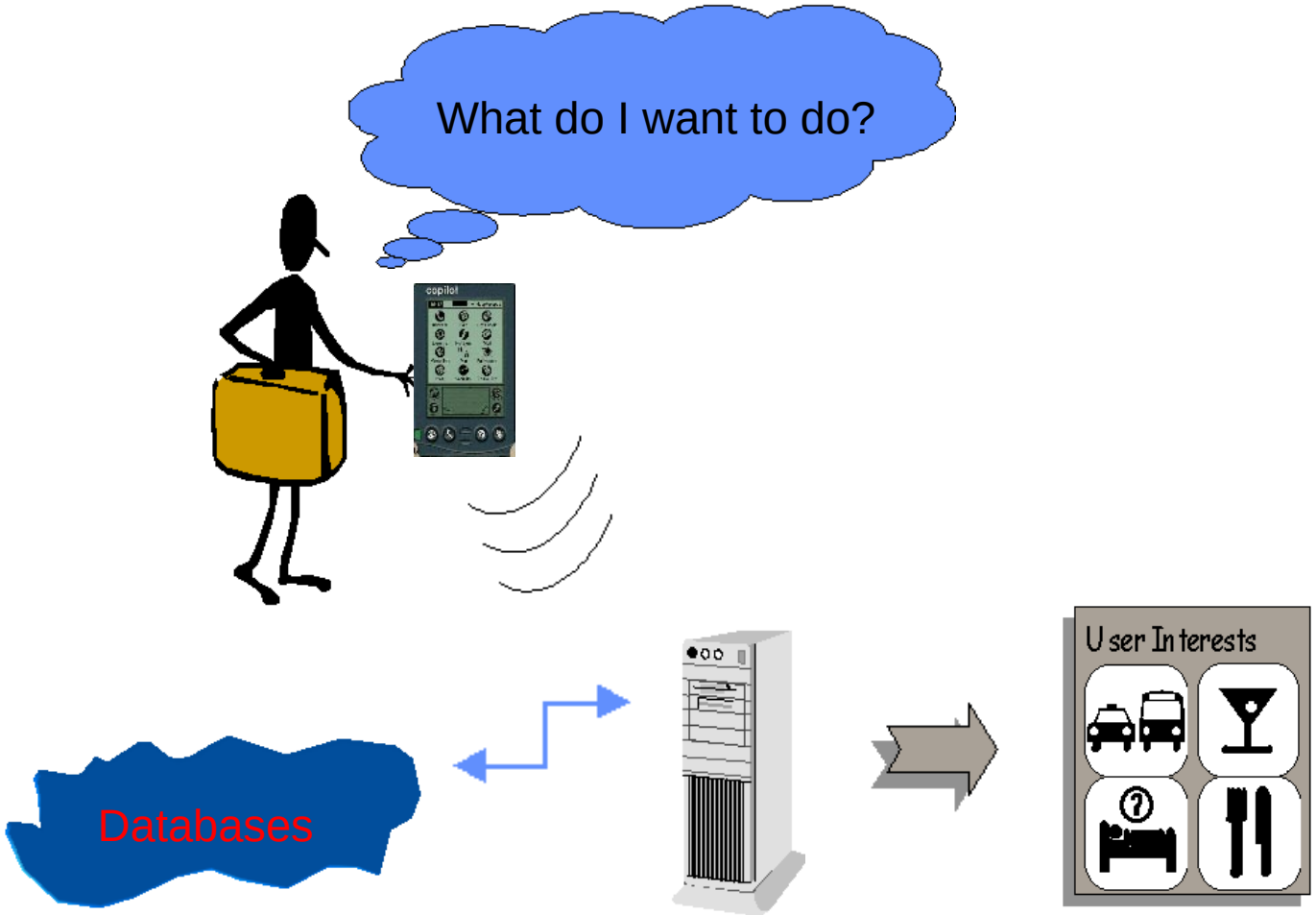
MIA — An agent based search engine

MIA — An agent based search engine

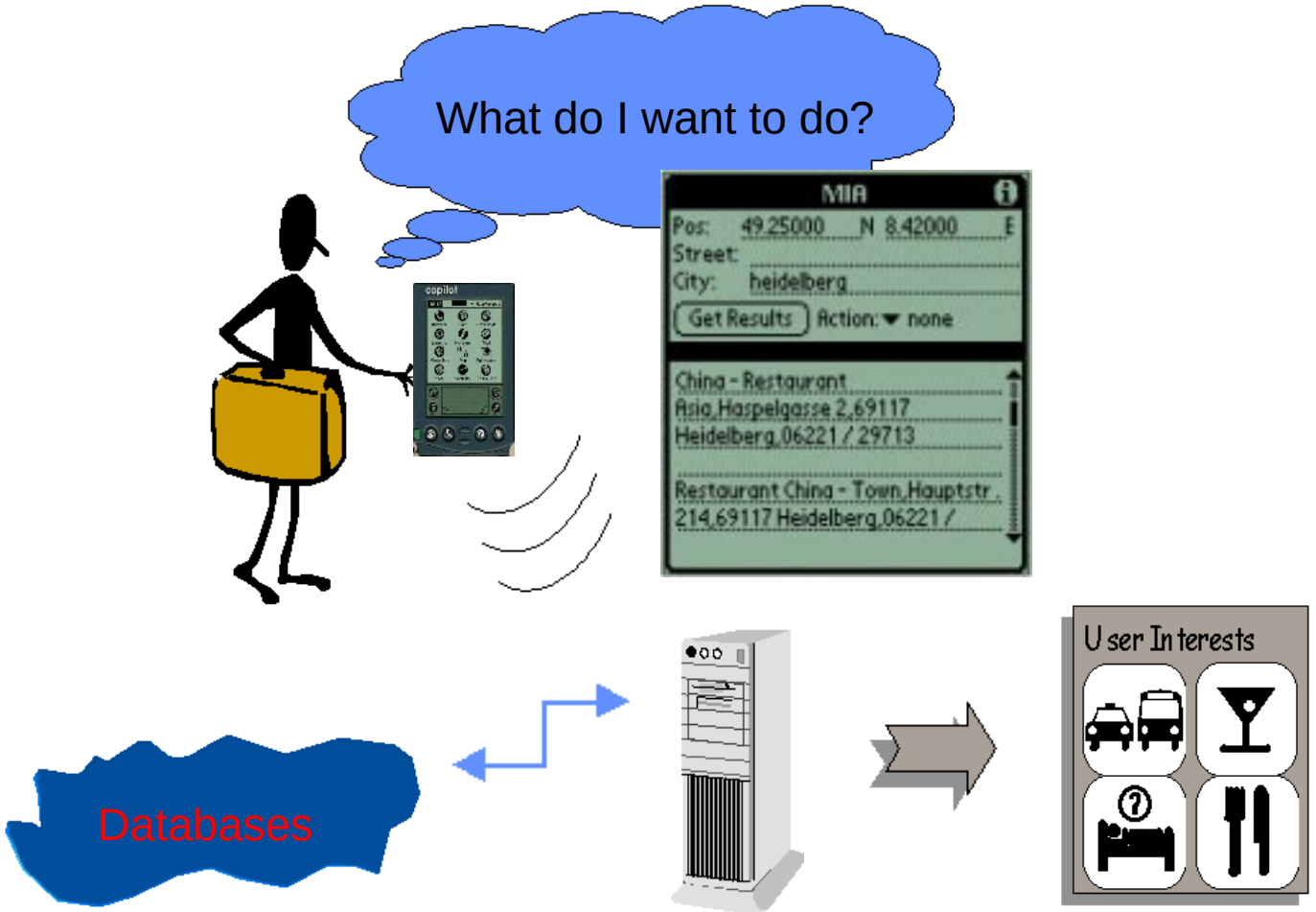
MIA — An agent based search engine



MIA — An agent based search engine



MIA — An agent based search engine

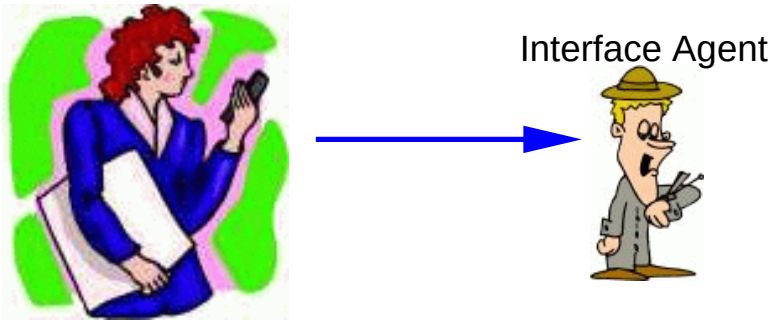


MIA Is Not An Ordinary Search Engine

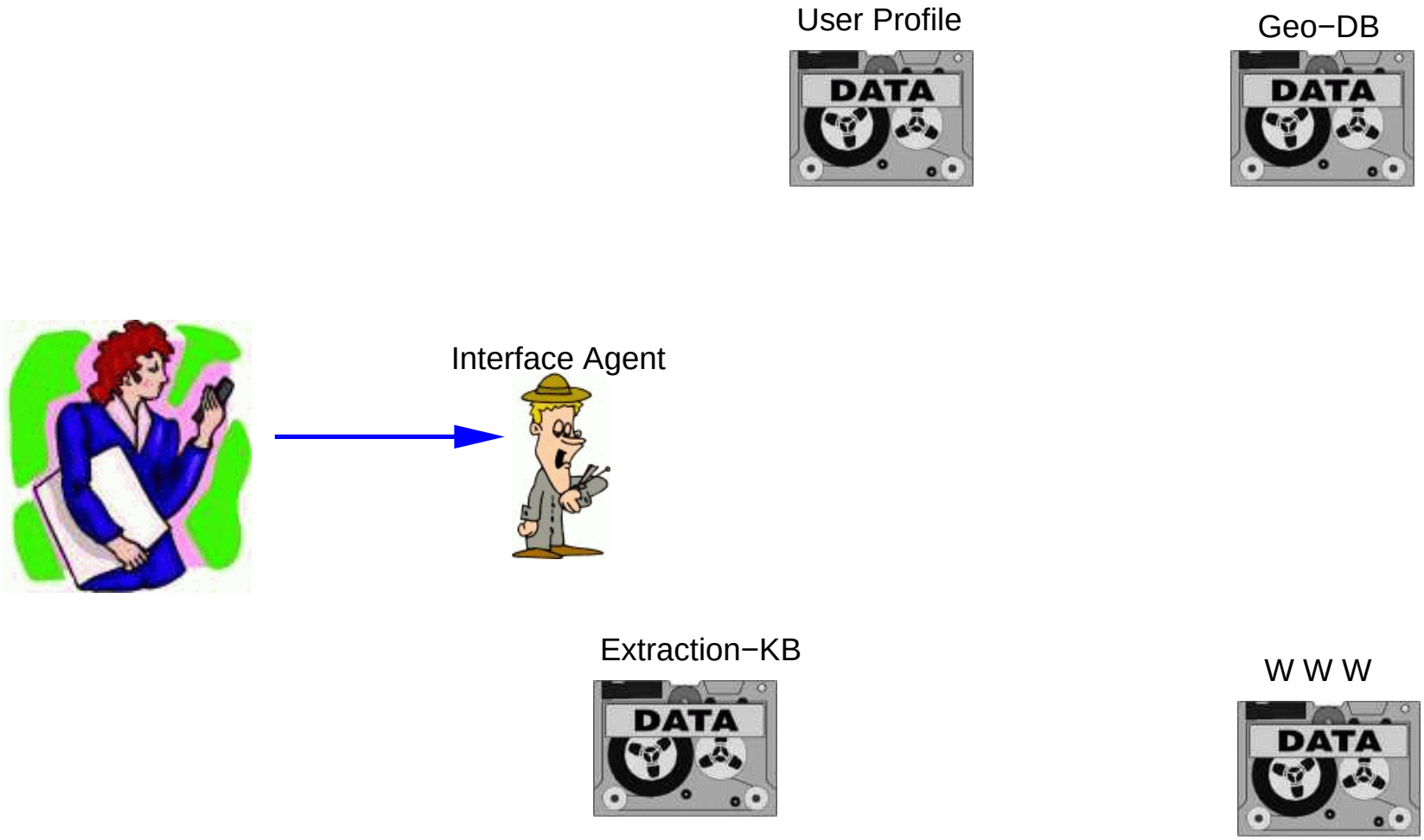
- Asynchronous Interactions
- Anytime algorithm
- Takes geographical data into account
- MIA provides information, not web pages
- AI / Agent-techniques

MIA's multi agent architecture

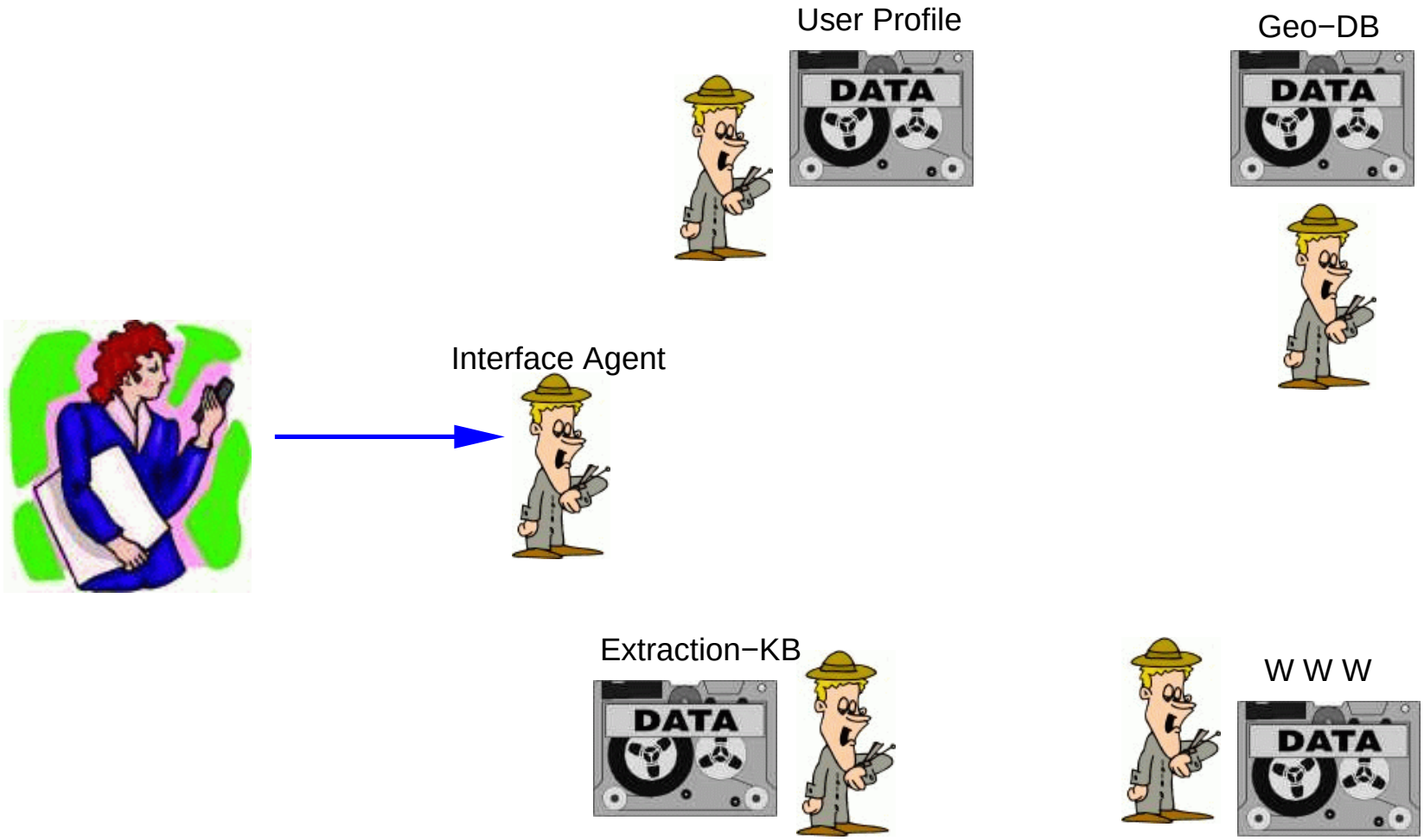
MIA's multi agent architecture



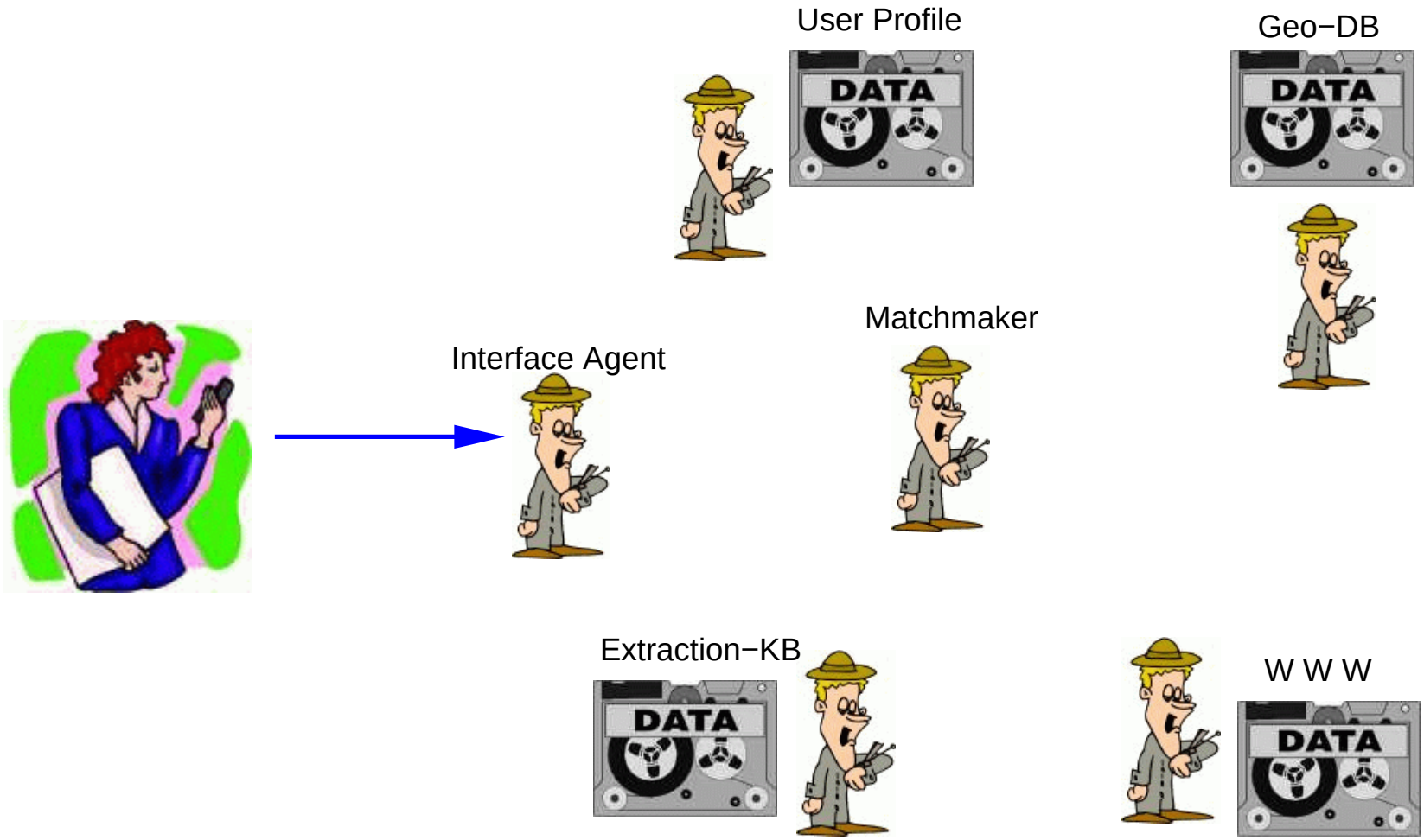
MIA's multi agent architecture



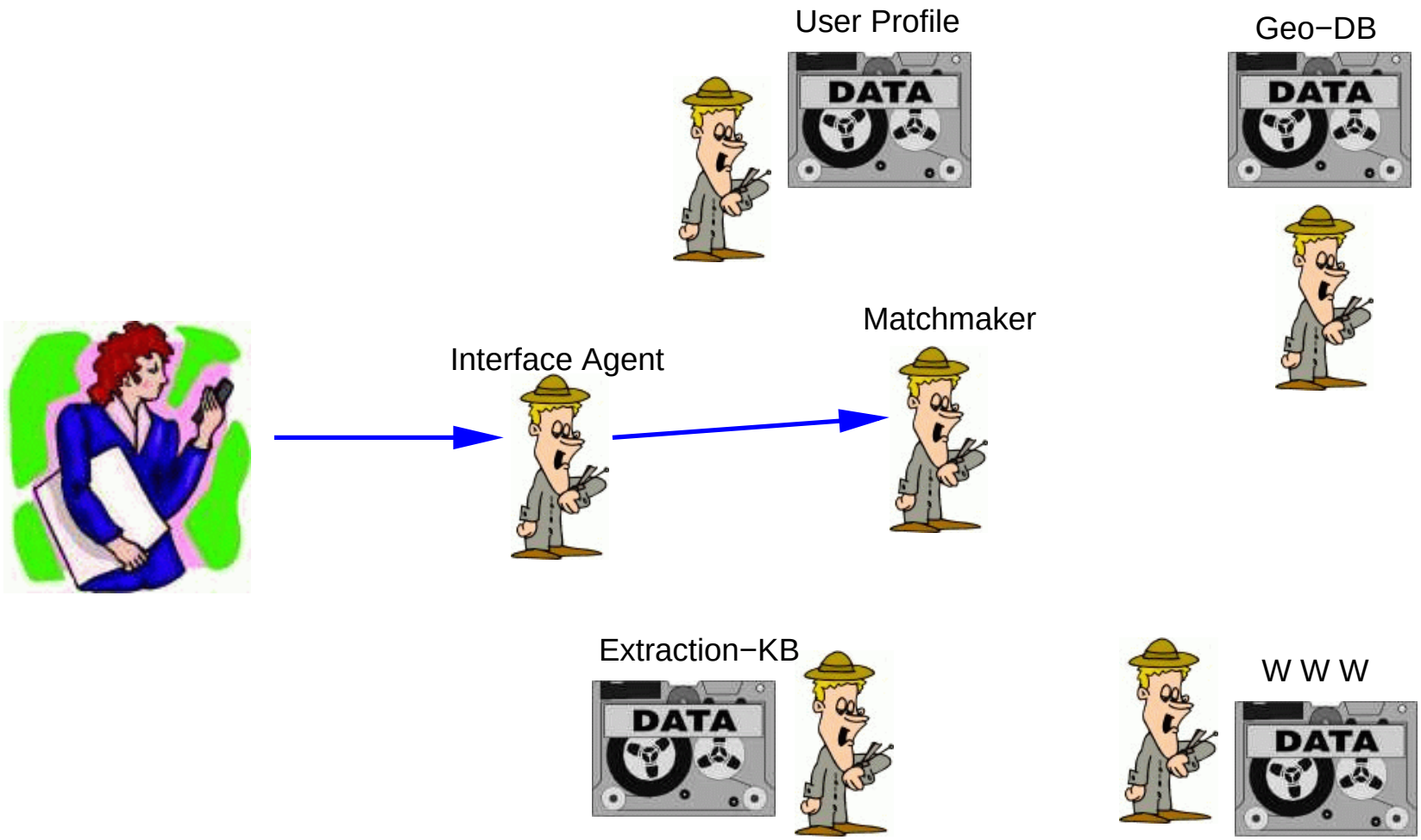
MIA's multi agent architecture



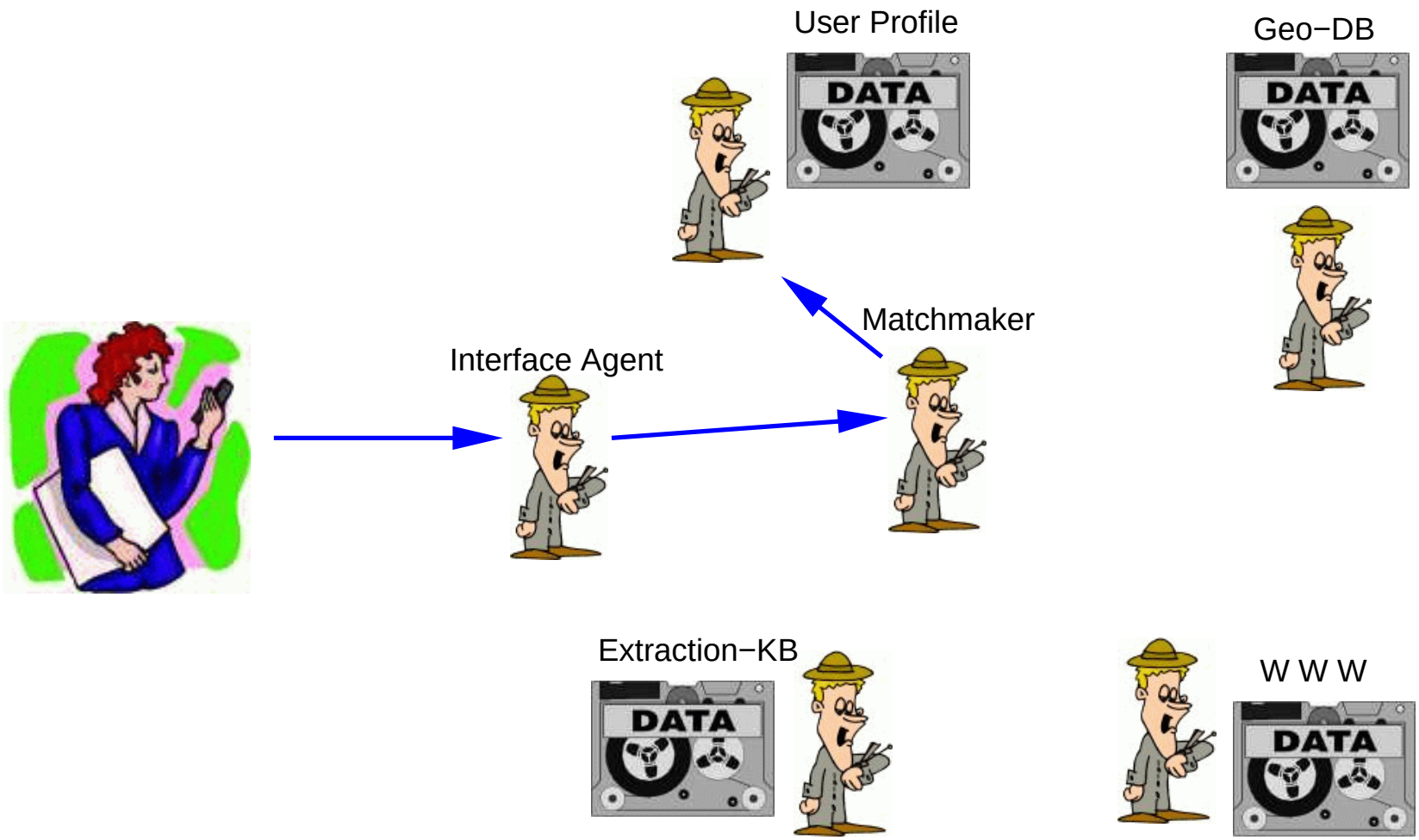
MIA's multi agent architecture



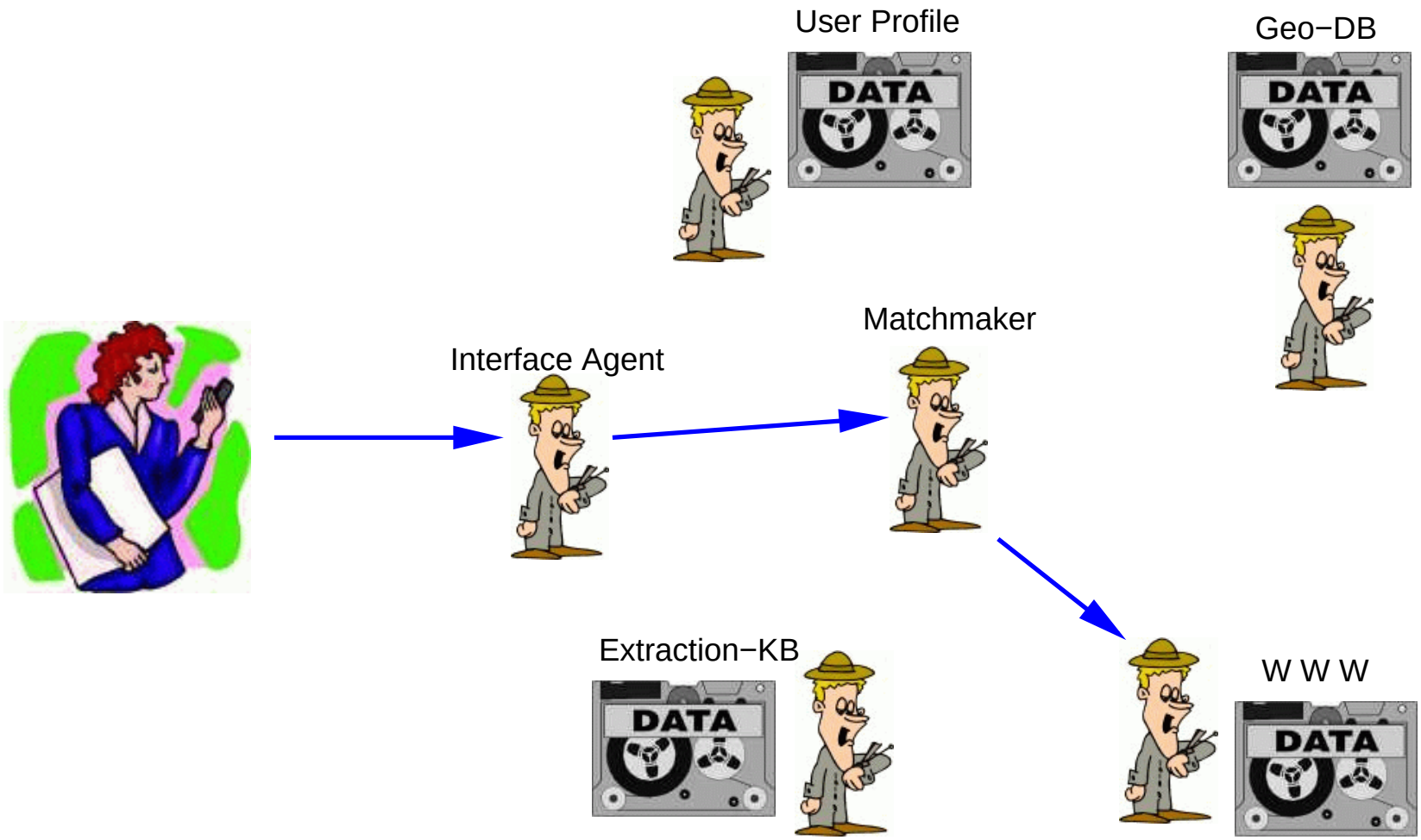
MIA's multi agent architecture



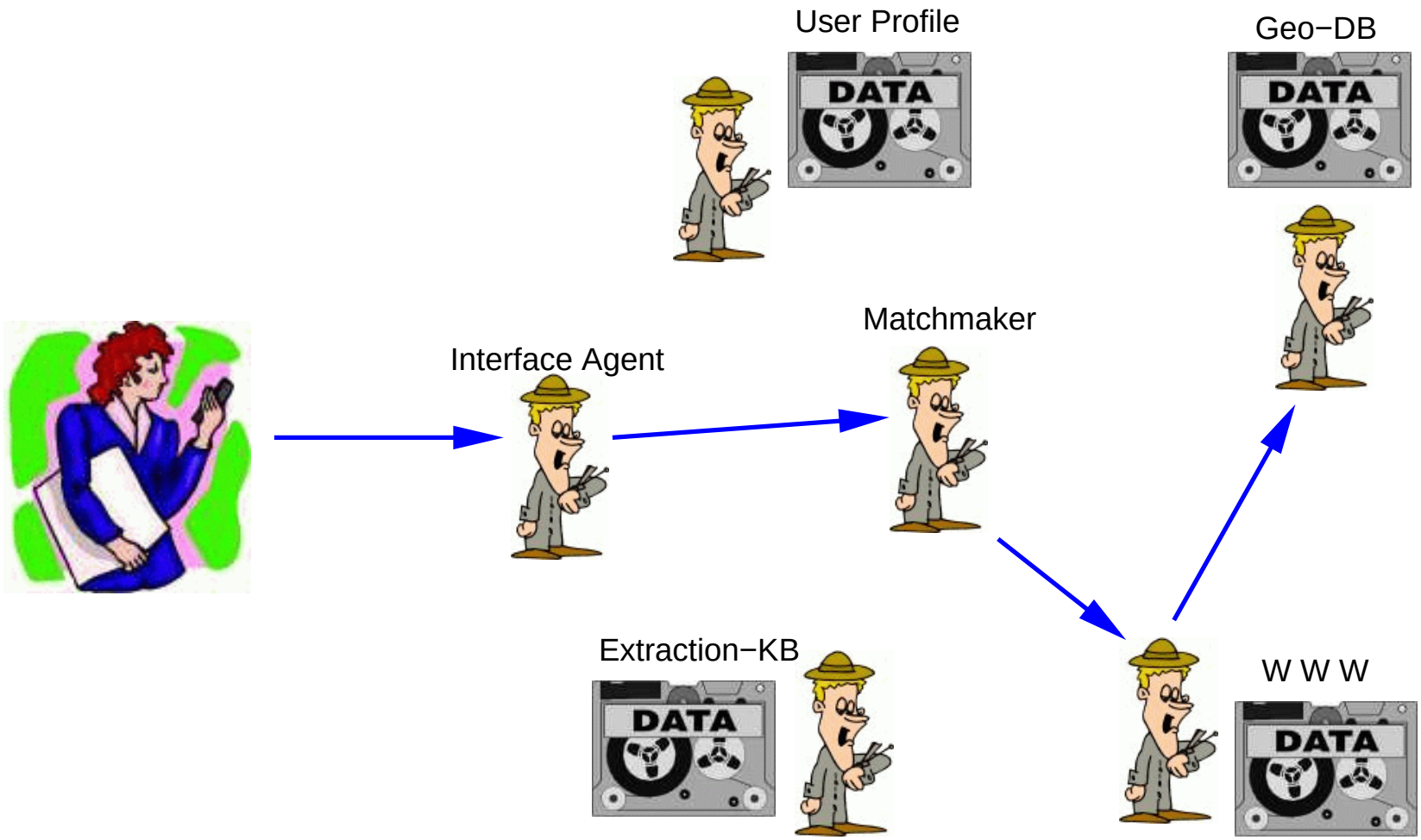
MIA's multi agent architecture



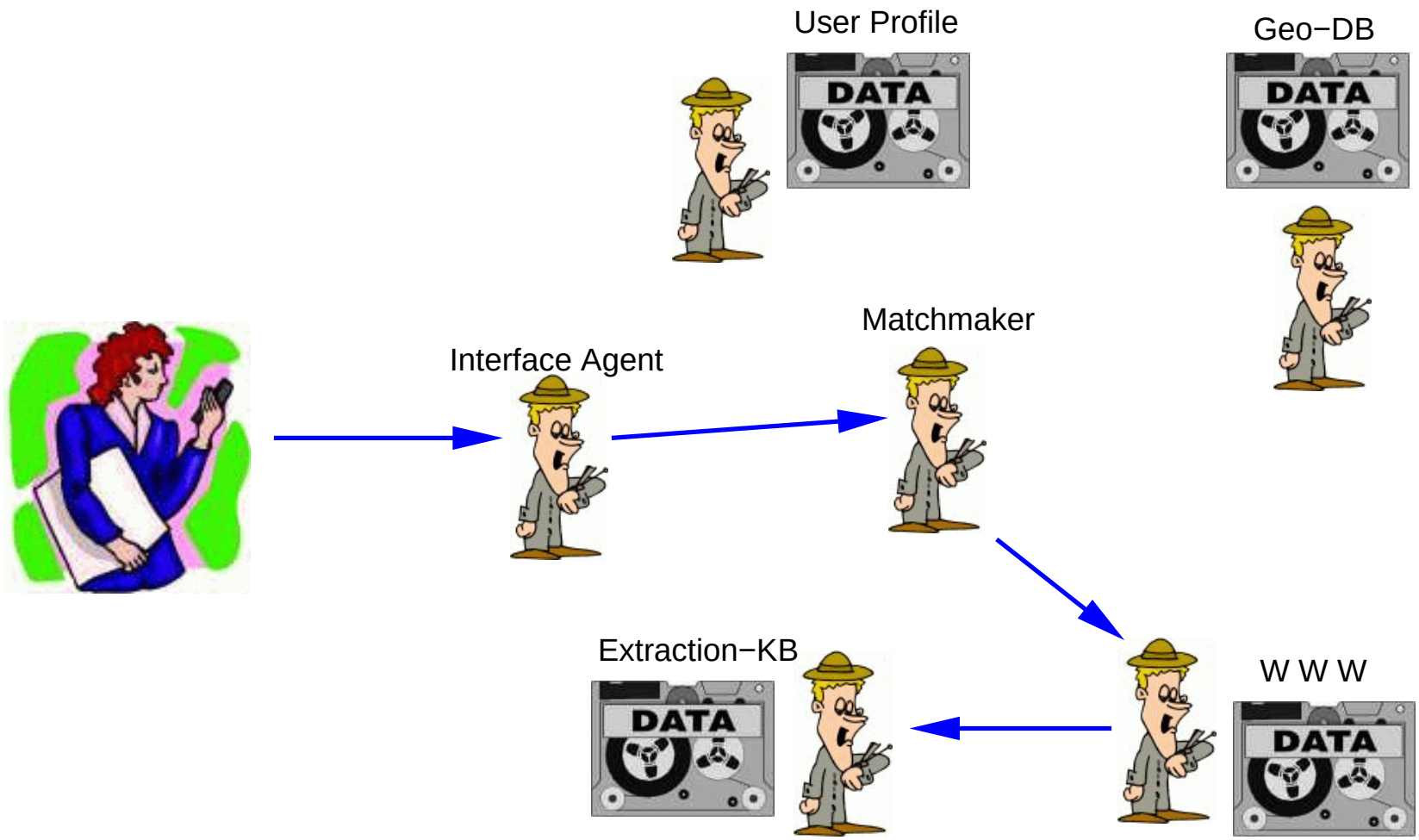
MIA's multi agent architecture



MIA's multi agent architecture



MIA's multi agent architecture



Accessing heterogenous databases

- One or more agents per database?
  - For simple databases one agent for all (GPS database)
  - For complex databases one agent per query (World Wide Web)

Accessing heterogenous databases

- MIA's task can be described as putting together information from various databases, e.g.
  - User's preferences
  - Web directories / yellow pages
  - Generic Internet search engines
  - Web Spiders
  - Geographical databases
  - Map servers
  - Ontologies
  - A priori unknown databases!

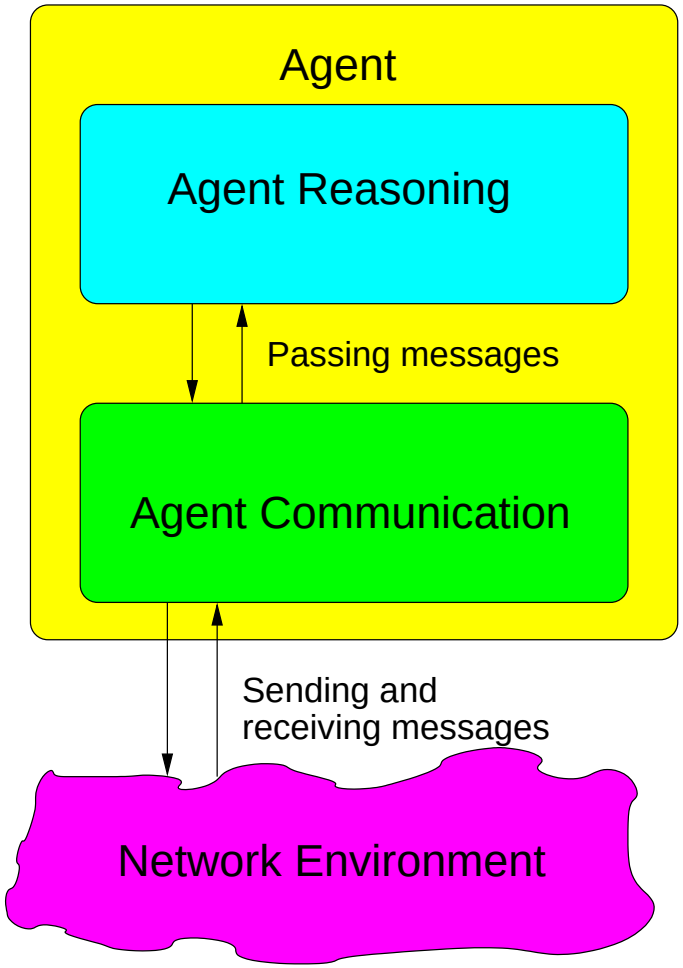
Logic agents

- *Logic agents*: Finding the right agent(s) and answering queries as construction of logical proofs.
- ```
  find(Topic, Type_of_information, Geographical_position,
       Extracted_information) :-
    query_agent(Matchmaker, 'geo_agent', Geo_agent),
    query_agent(Geo_agent, Geographical_position, [Street, City]),
    query_agent(Matchmaker, 'spider_agent', Topic, Topic_agent),
    query_agent(Topic_agent, Topic, [Street, City], Potential_answer),
    query_agent(Matchmaker, 'extraction_agent',
                Type_of_information, Extraction_agent),
    query_agent(Extraction_agent, Type_of_information,
                Potential_answer, Extracted_information).
  ```
  ```

Heterogenous agents' architecture

- Matchmaker agent as the main facilitator
- Agents who want to delegate tasks to other agents ask matchmaker agent
- Not limited to this setup: An agent can decide on its own how to solve task and which agents to contact
- Goal:
  - When a MIA agent gets introduced to an unknown information agent, it should be able to communicate with the new agent, thereby enhancing the MIA system.
  - This requires a common language!

Agent communication — Library



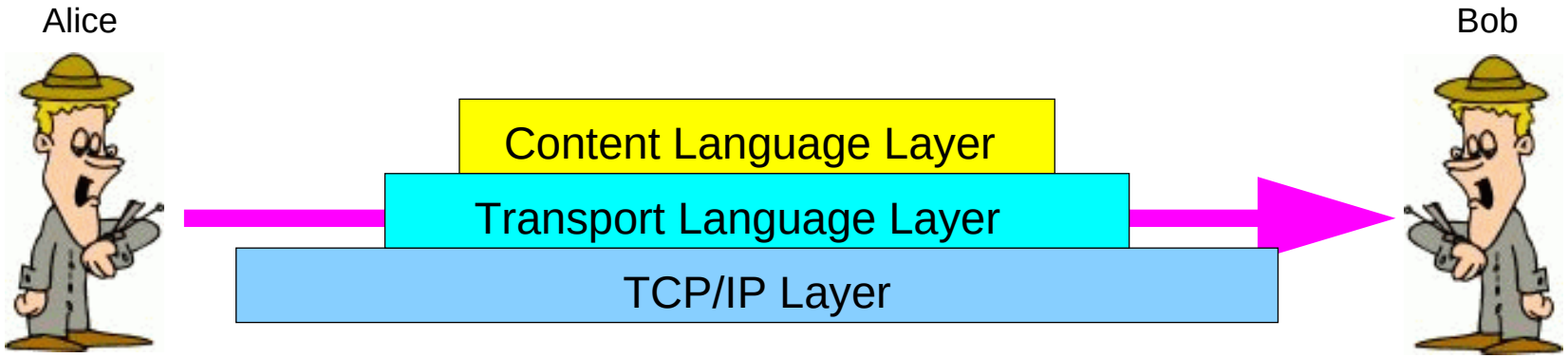
- Library takes care of sending, receiving and parsing of messages
- Alpha-version for Eclipse PROLOG in test right now.
- In the future, it will be available (at least) for other varieties of PROLOG as well.

Agent communication — Language

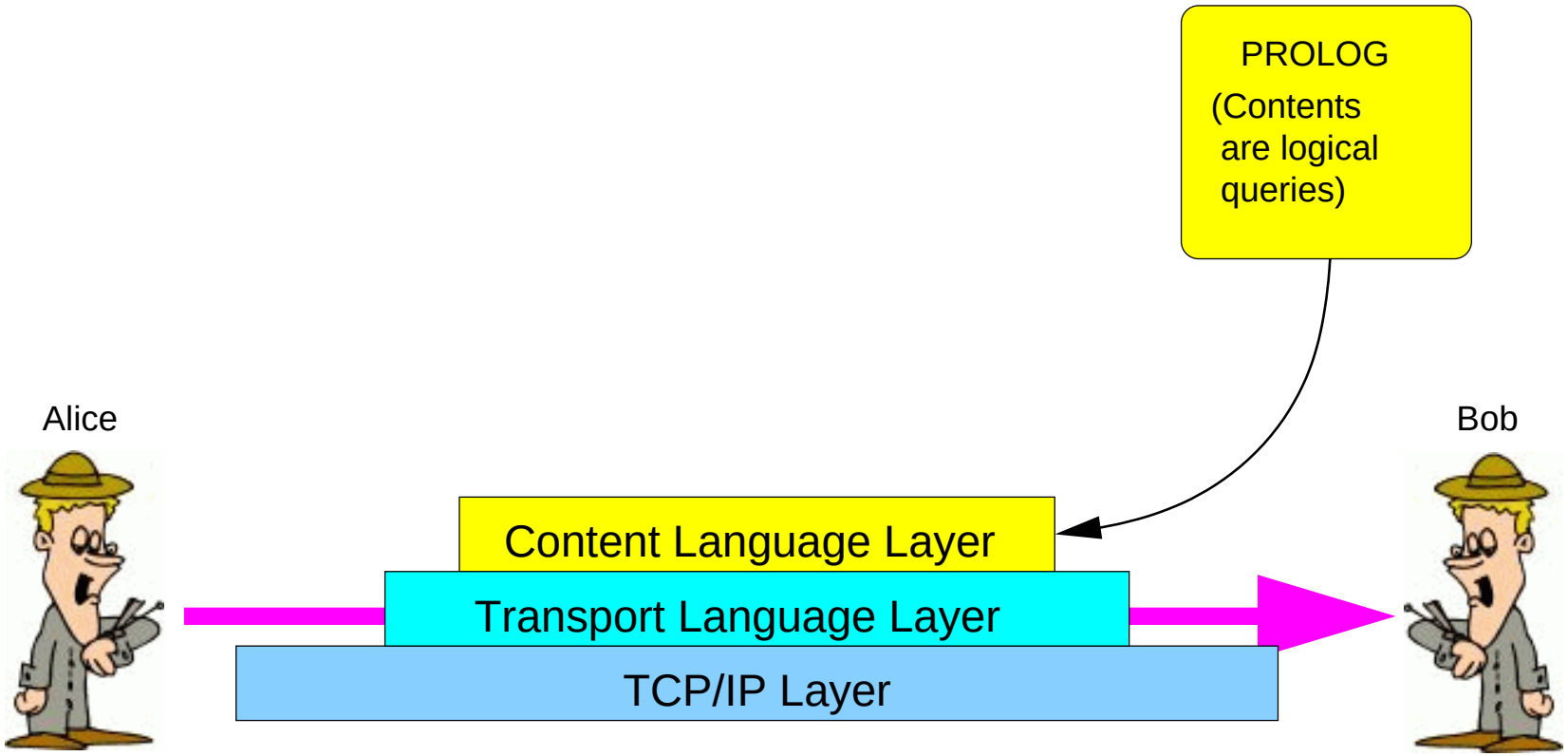
- Goal: Allow communication with alien (unknown) agents
- Use standardized communication model
- $\Longrightarrow$ KQML

Agent communication — Layers

Agent communication — Layers



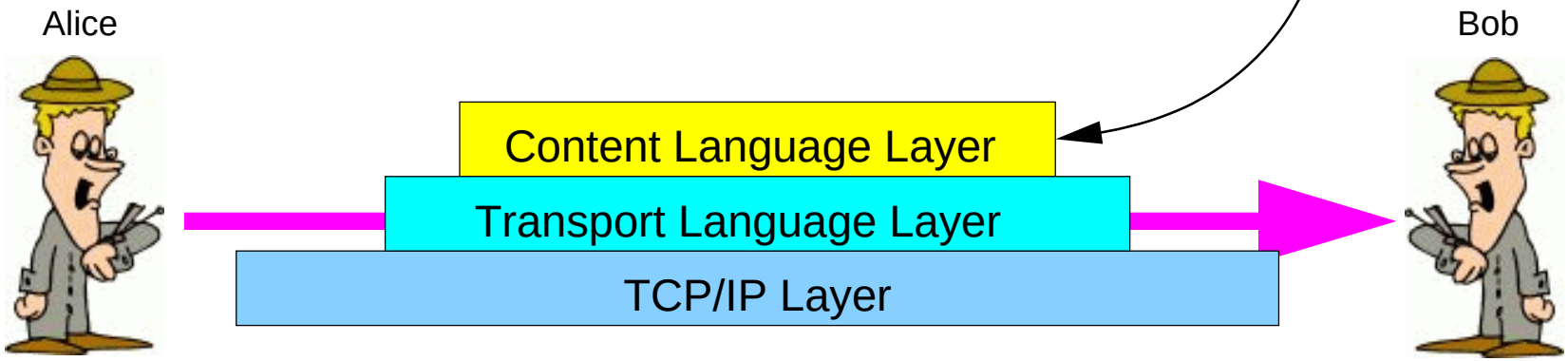
Agent communication — Layers



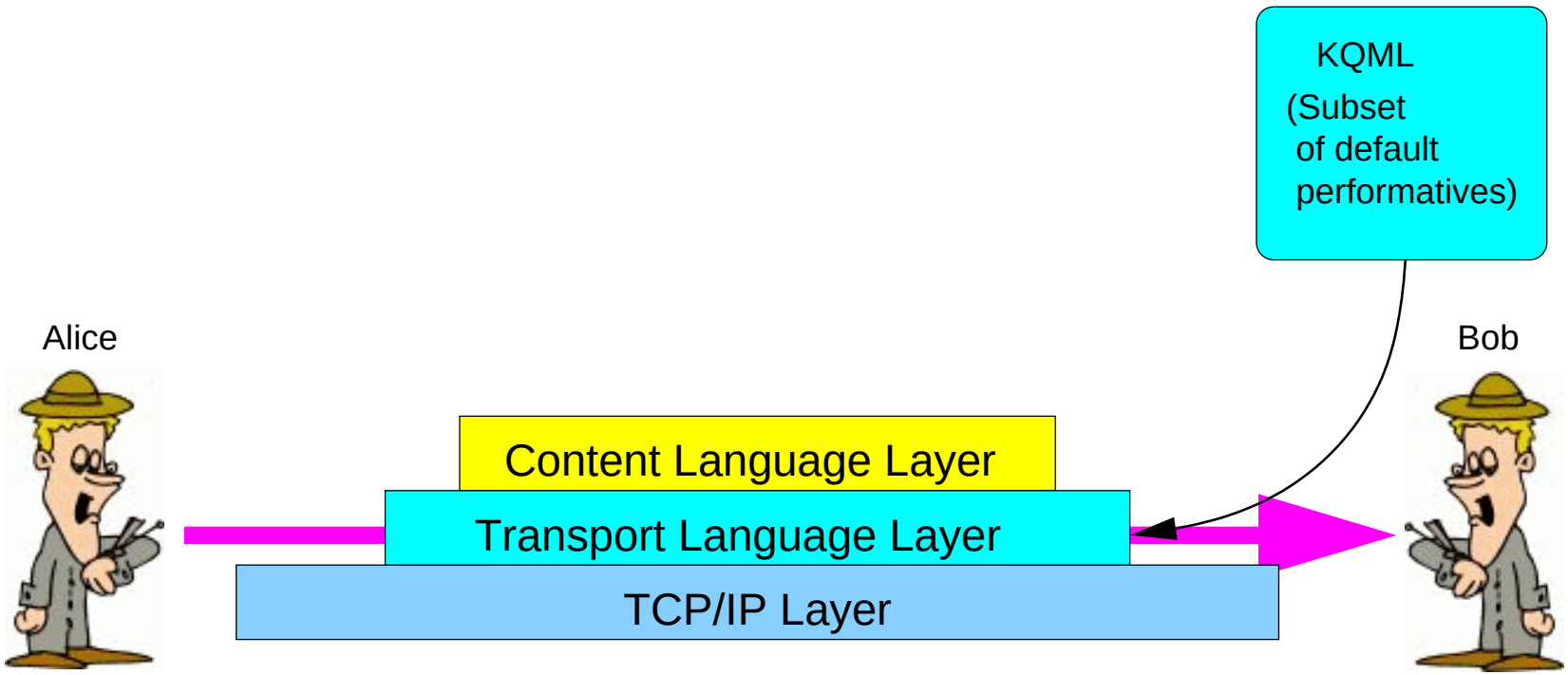
Agent communication — Layers

```
spider_result('Cologne',
              'Addresses',
              'Restaurants',
              Result).
```

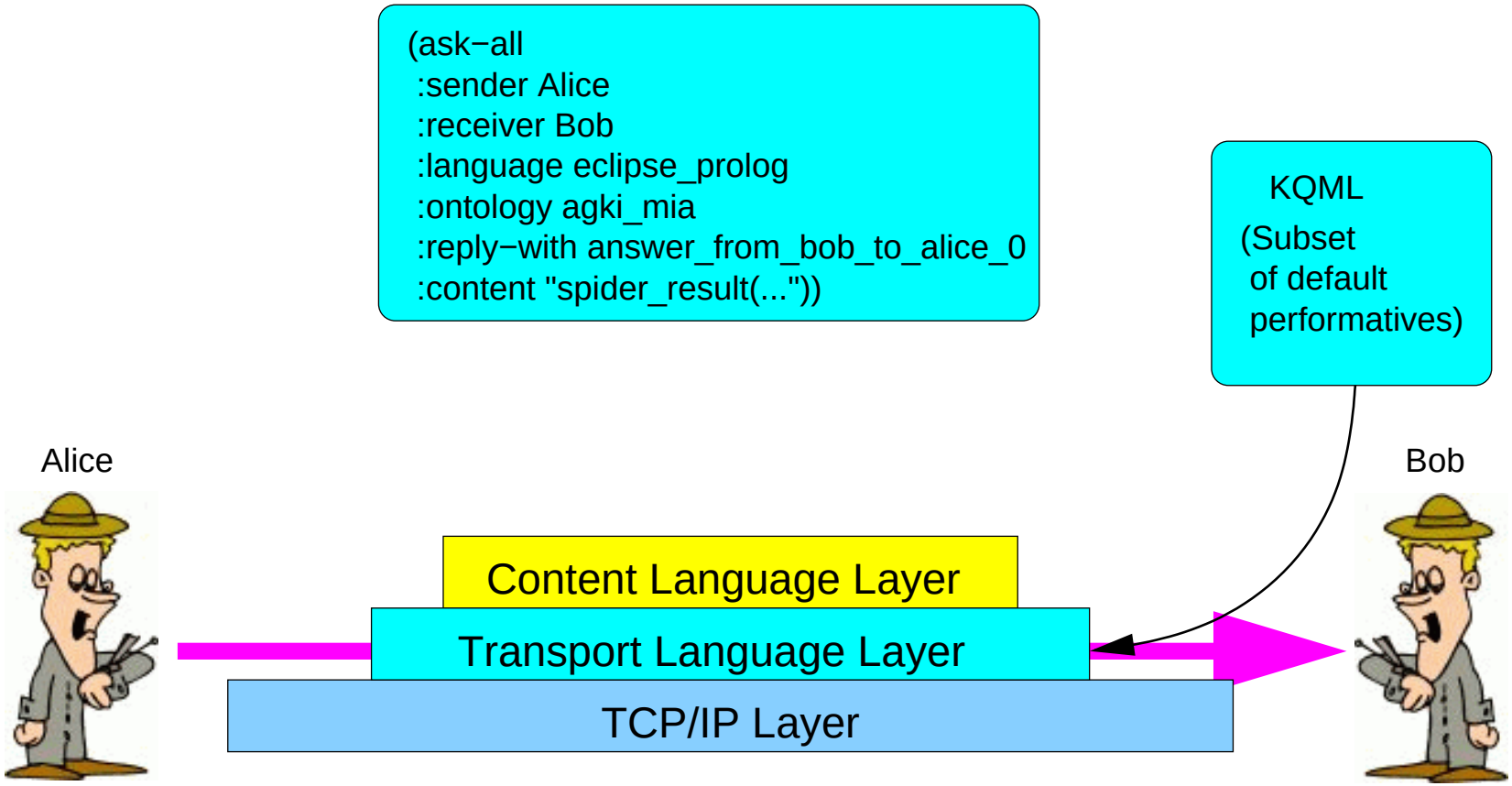
PROLOG
(Contents are logical queries)



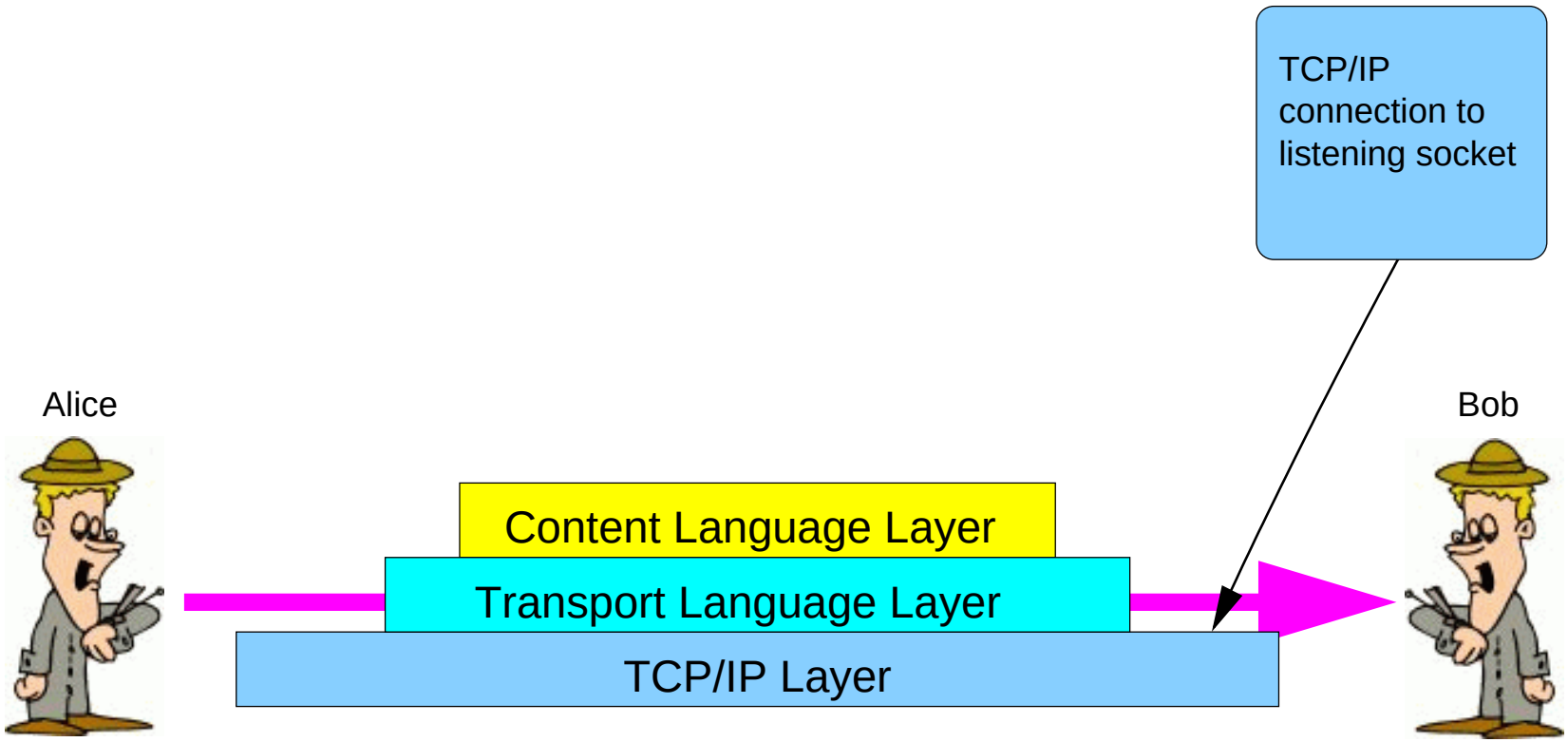
Agent communication — Layers



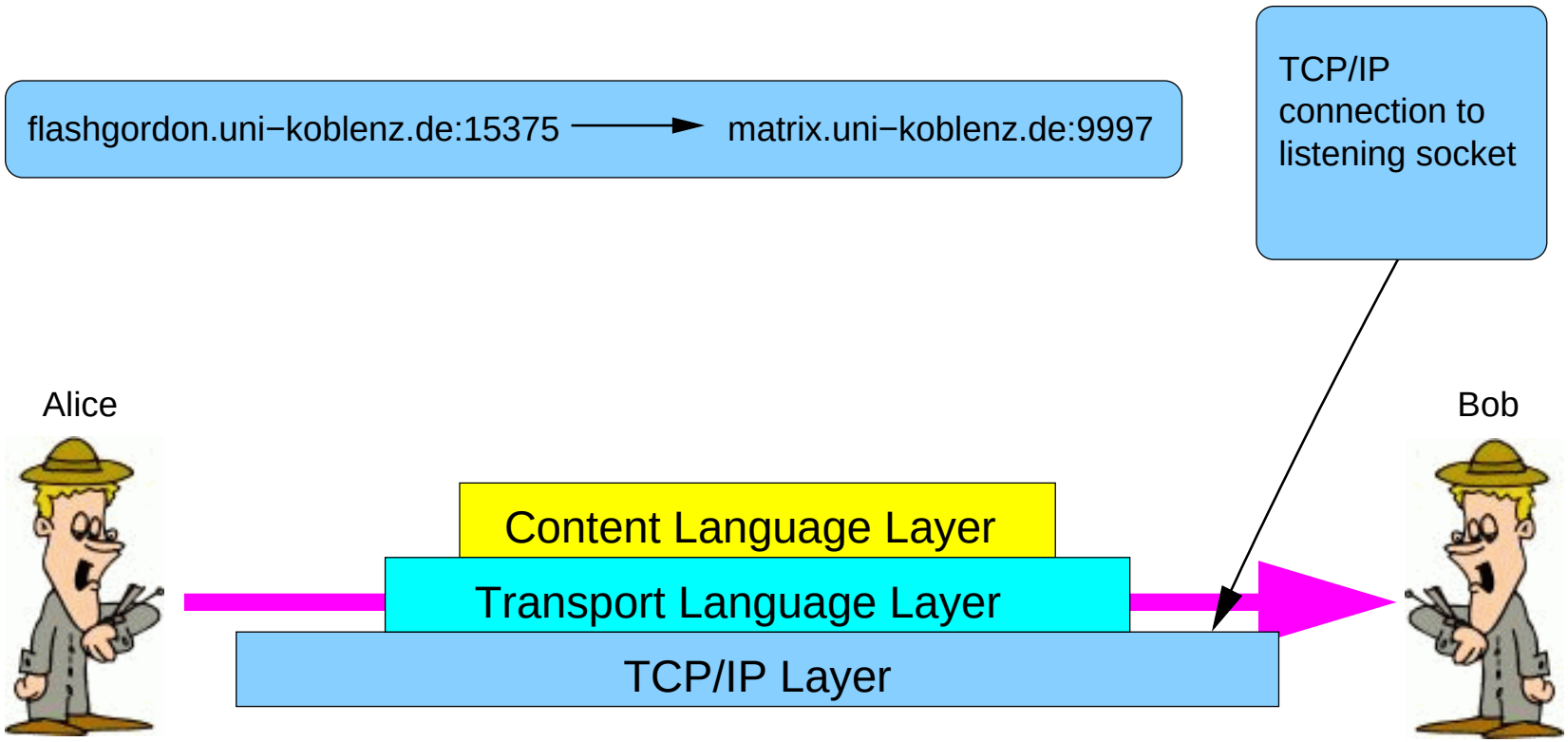
Agent communication — Layers



Agent communication — Layers



Agent communication — Layers

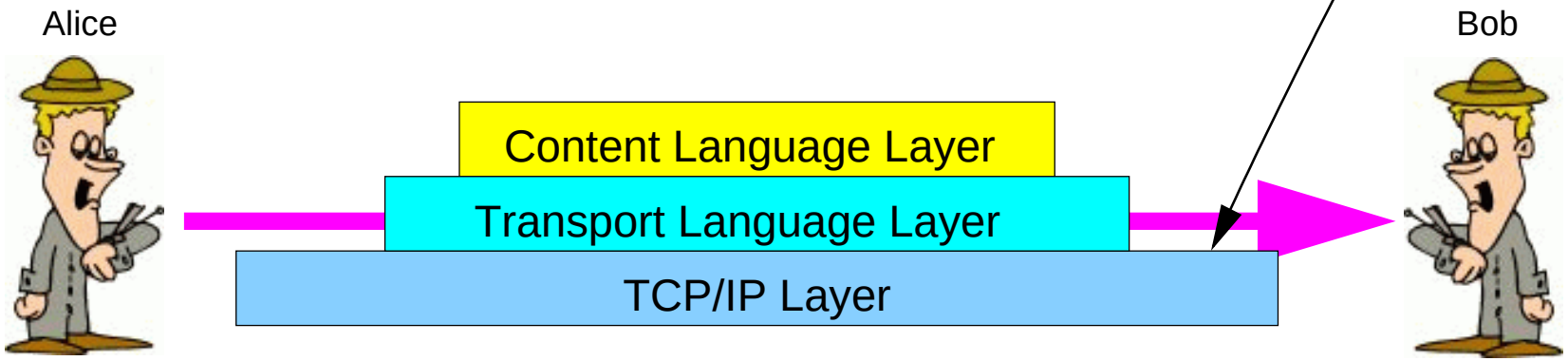


Agent communication — Layers

Problem: Sending port not identical to listening port!

flashgordon.uni−koblenz.de:15375 → matrix.uni−koblenz.de:9997

TCP/IP connection to listening socket



Agent communication problems — Transportation layer

- KQML definition not sufficient for TCP/IP communication:
- Just by receiving a message via TCP/IP, the receiving agent does not know how to contact the sender.
- Solutions:
  - Encode agent's location in its name
  - DNS-like resolver agent
  - Additional parameters in message
- No interoperability between KQML implementations!
- Violates the standard!

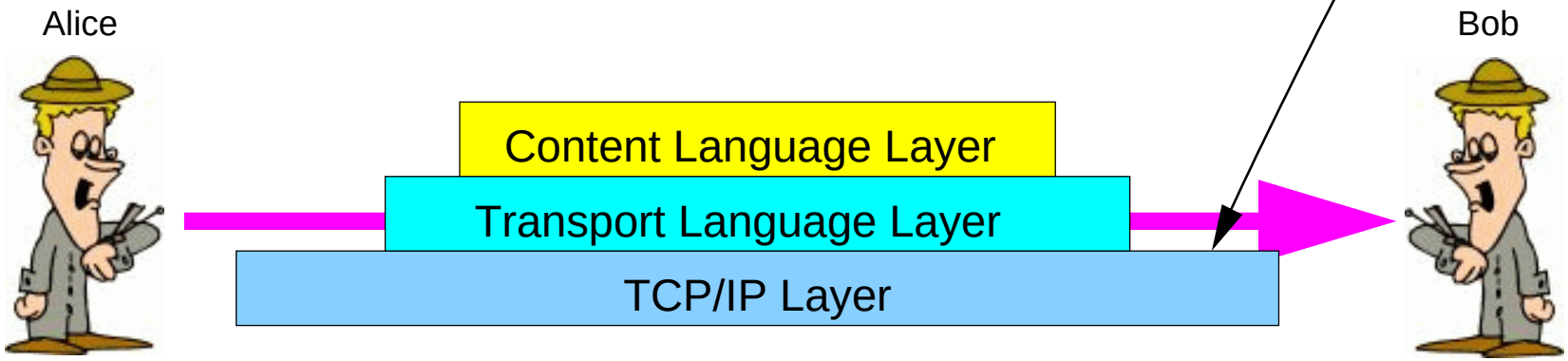
Agent communication — Layers

Problem: Sending port not identical to listening port!

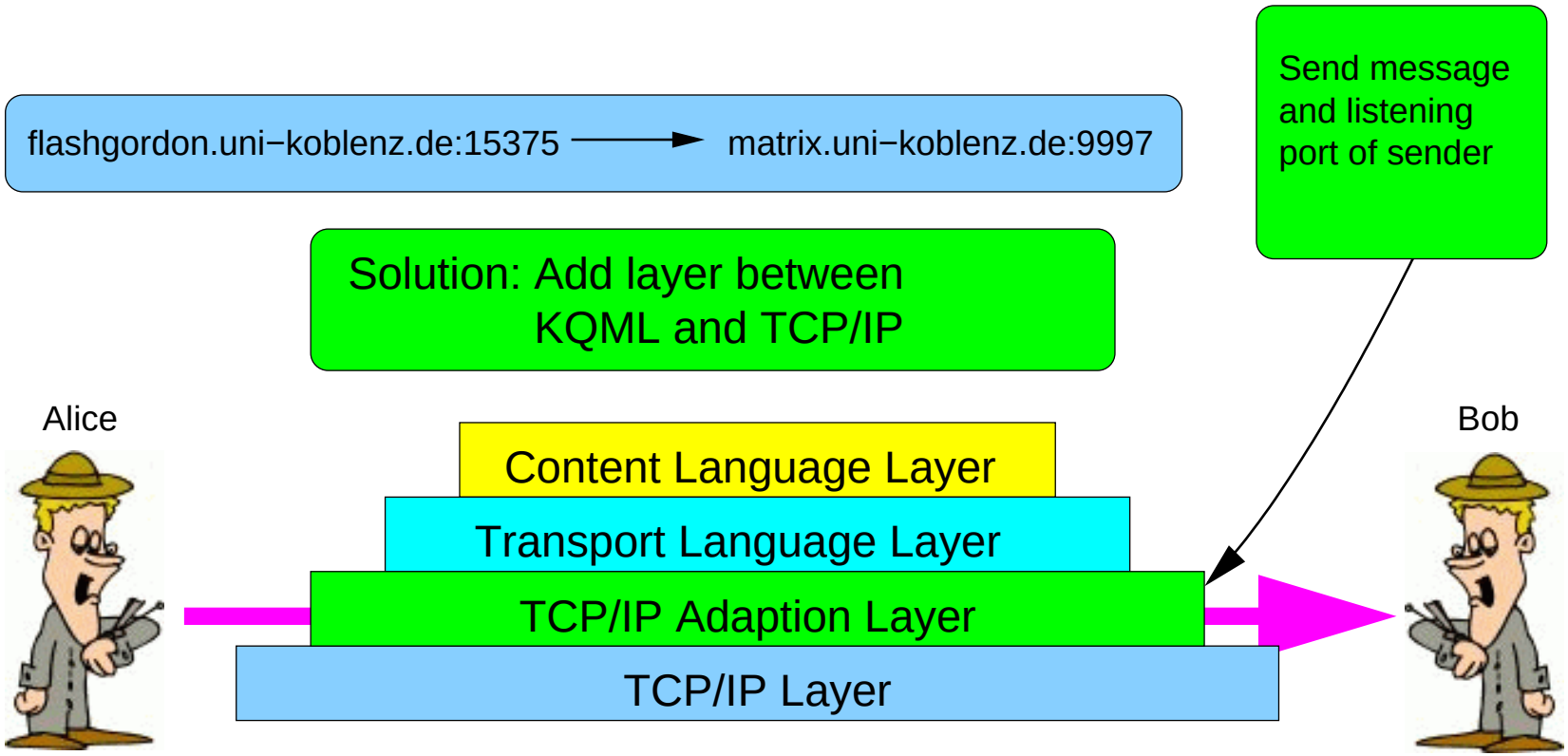
flashgordon.uni−koblenz.de:15375 → matrix.uni−koblenz.de:9997

TCP/IP connection to listening socket

Solution: Add layer between KQML and TCP/IP



Agent communication — Layers



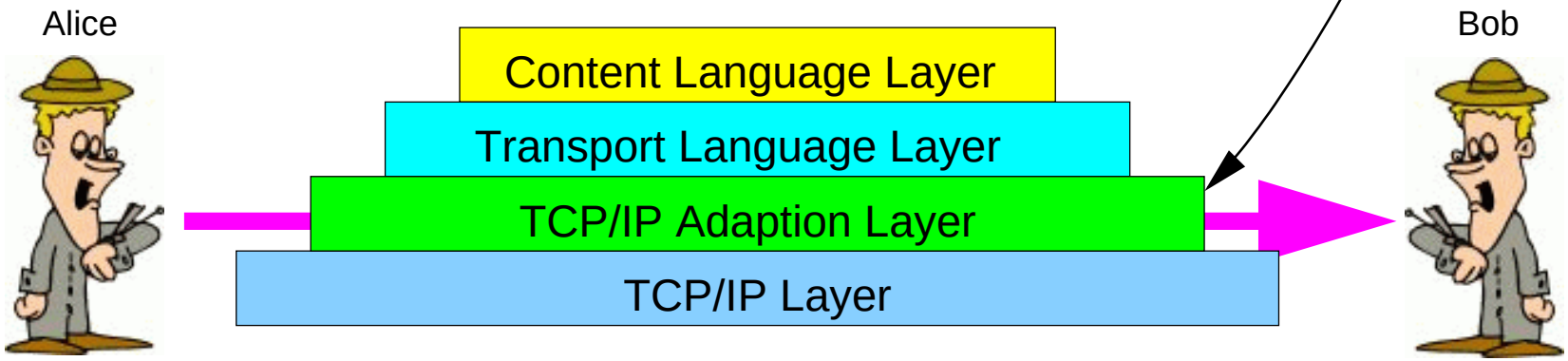
Agent communication — Layers

flashgordon.uni−koblenz.de 10000 (ask−all :sender Alice :receiver Bob ...

flashgordon.uni−koblenz.de:15375 ⟶ matrix.uni−koblenz.de:9997

Send message and listening port of sender

Solution: Add layer between KQML and TCP/IP



Agent communication

- In order to communicate, the agents must agree not only on a communication protocol, but also on a common language and ontology for the messages's content
- Right now, we use (Eclipse-)PROLOG as the language, and a simple, implicit ontology.

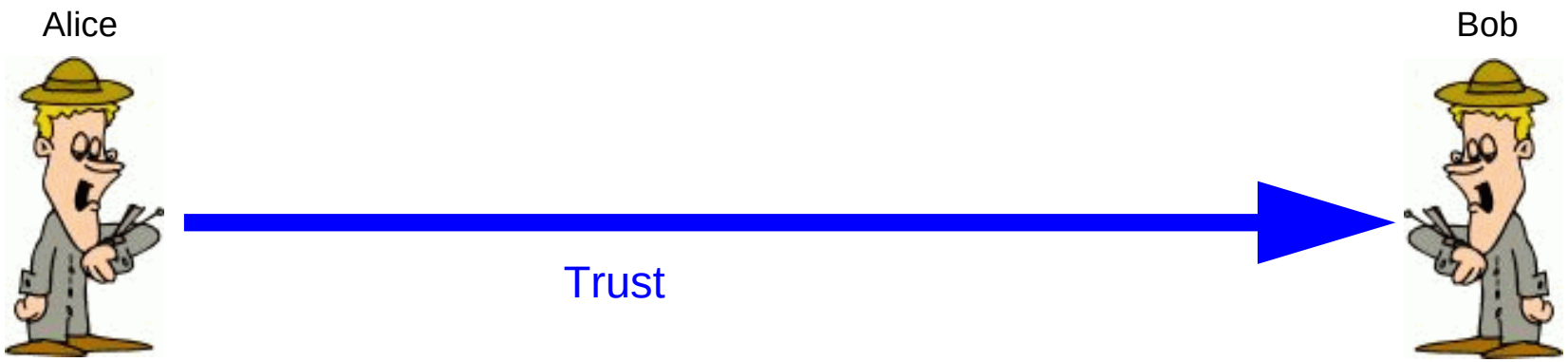
Critique of KQML

- Performatives and content are not clearly distinguished.
- Examples:
  - `(ask-all :content "agent_results(X)")` $\equiv$
    `(ask-one :content "findall(Y, agent_result(Y), X)"`
  - `(achieve :content "foo(bar)"` $\equiv$
    `(ask-if :content "foo(bar)")`
  - `(insert :content "foo(bar)")` $\equiv$
    `(ask-if :content "asserta(foo(bar))")`
- $\Rightarrow$ Security!

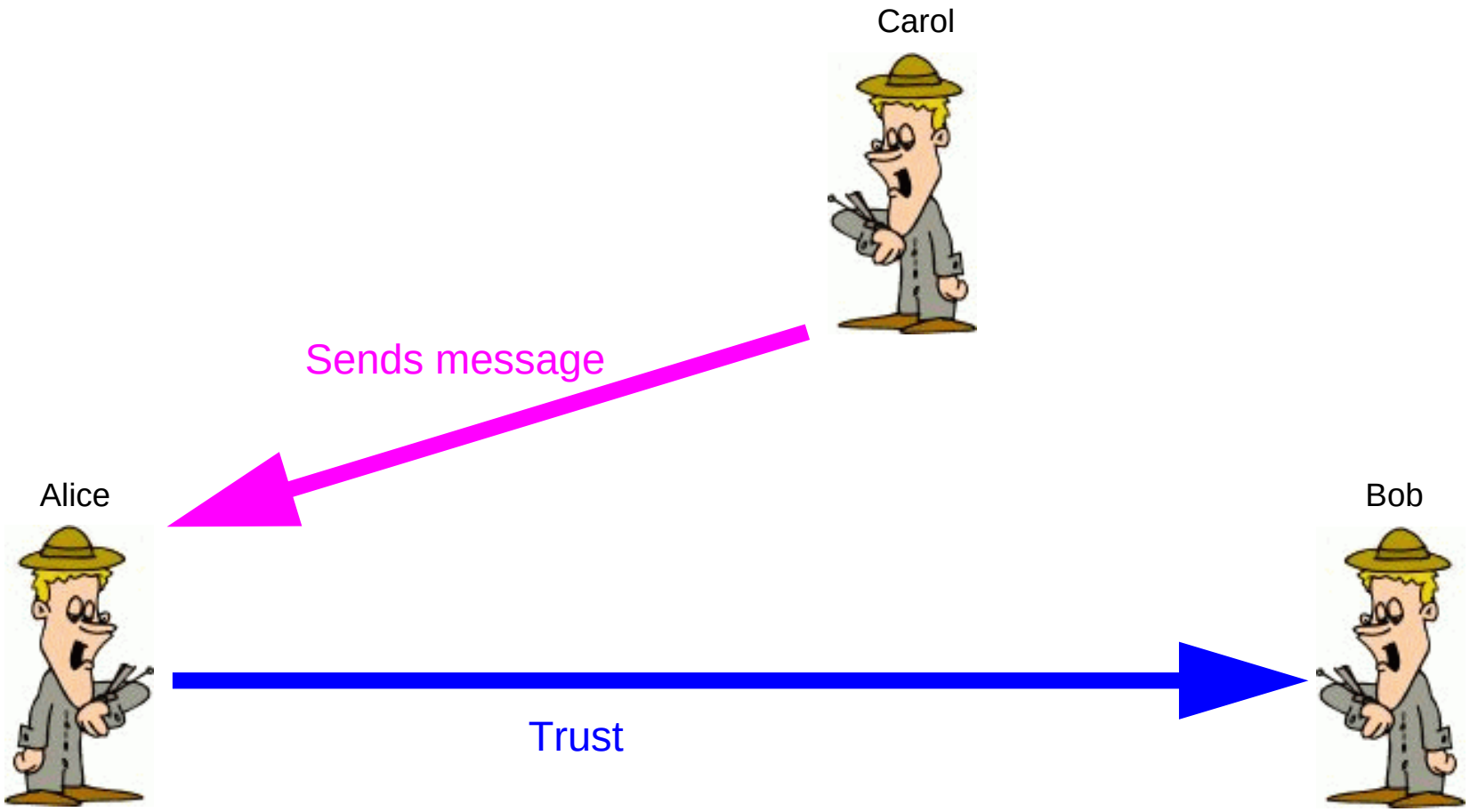
Interacting with alien agents — Security

- In Prolog, it is hard to limit the interface
- Example:
  - `(ask-if :content "asserta(foo(bar))")`
- How to deal with agents serving malicious data? ("Spam agent" redirect queries for cinemas, cultural events, etc.)
- Authentification / Limited access
  - MIT-Magic-Cookies
  - Public-Key Encryption
  - Problem: How to deal with unknown agents?

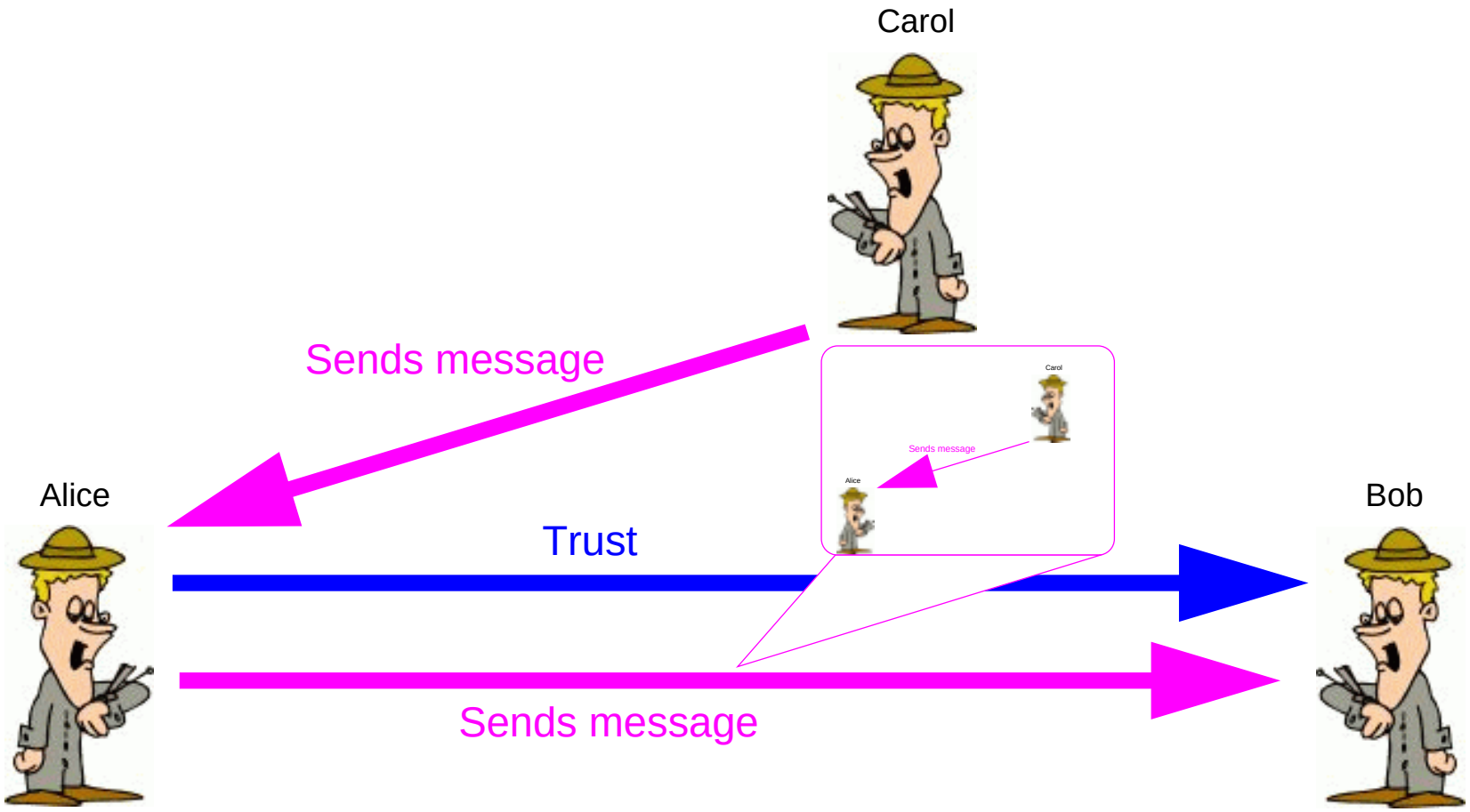
Web of Trust



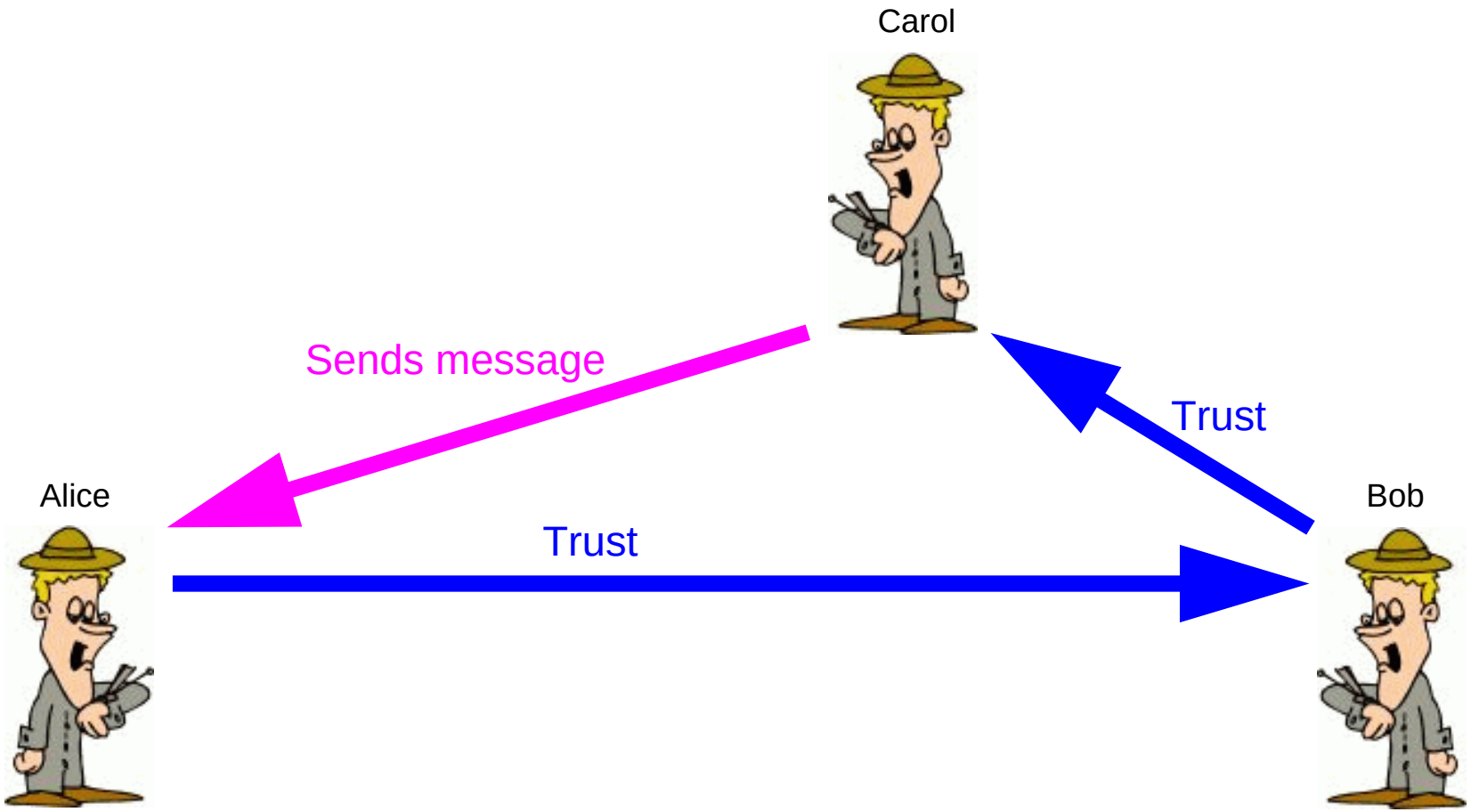
Web of Trust



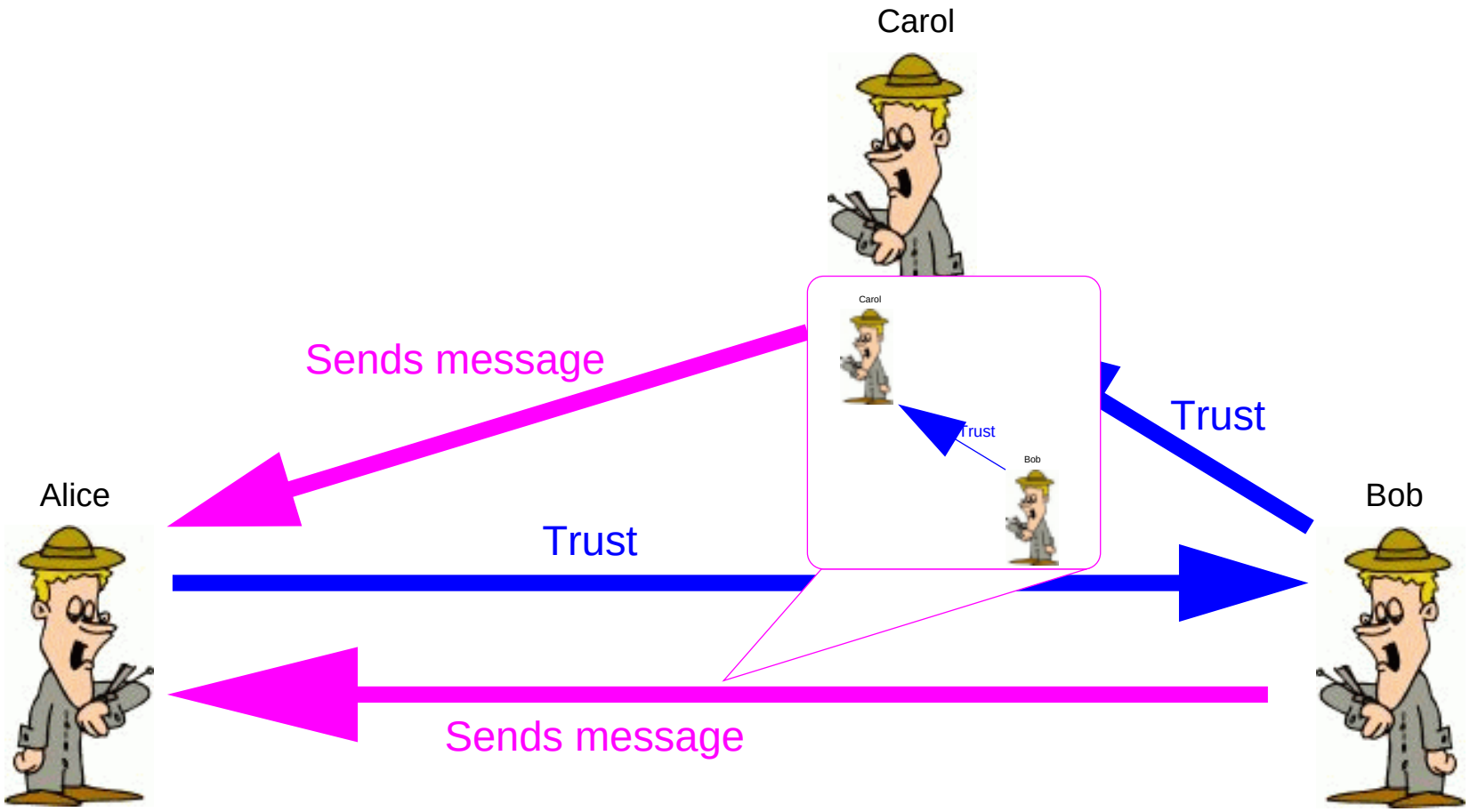
Web of Trust



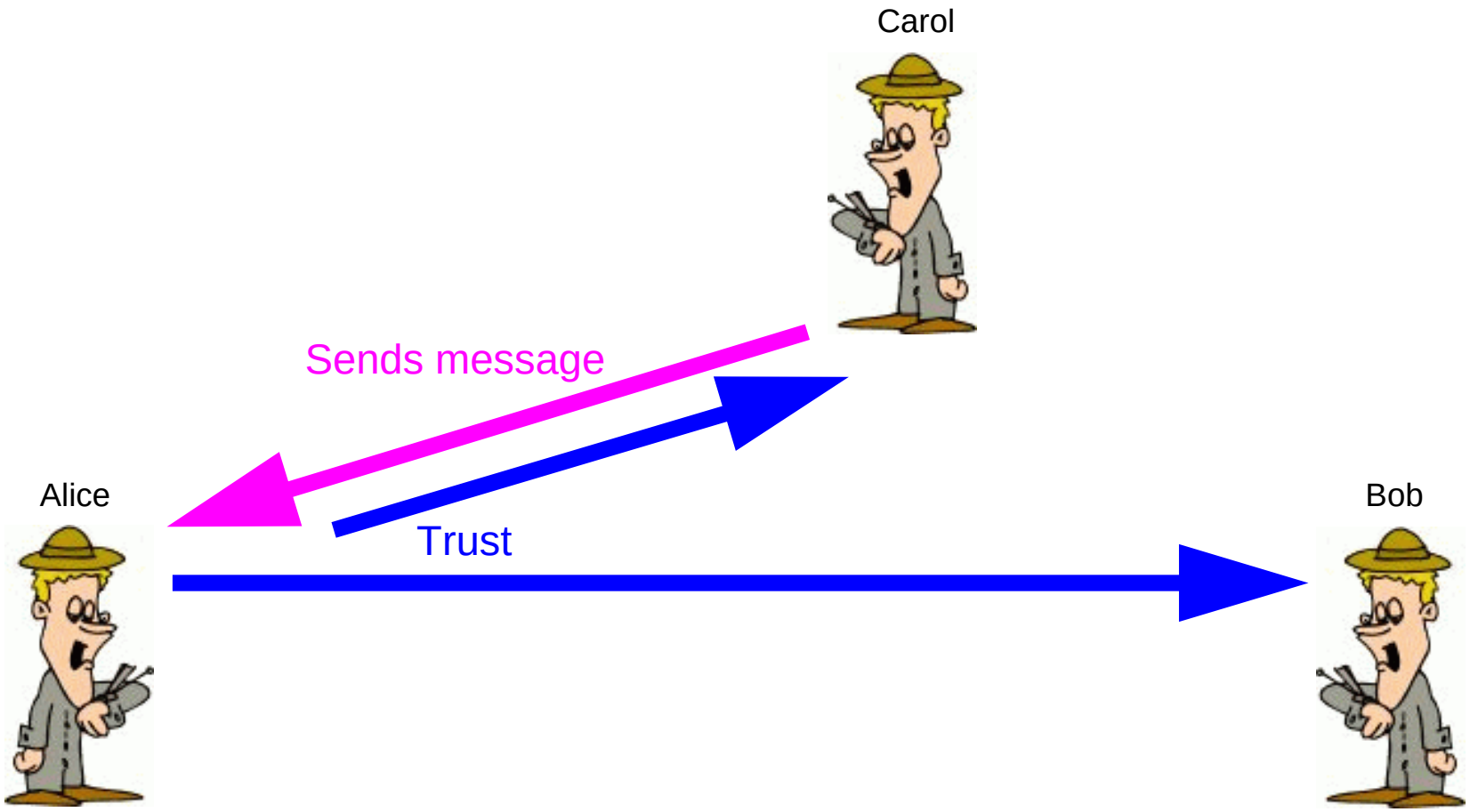
Web of Trust



Web of Trust


Carol
Carol
Trust
Bob
Sends message
Trust
Alice
Trust
Bob
Sends message

Web of Trust



Agent communication — Language

- KQML is independent of language
- For flexibility, support for multiple language is desirable.
- Beside PROLOG (used right now), we intend to support KIF
- Since KIF = first order predicate logic plus extentions, it can easy be transfer to PROLOG.

Agent communication — Ontology

- So far, MIA uses a simple implicit ontology.
- Lack of generic ontologies
- In order to share knowledge with alien agents, different ontologies should be understood.
- Problem to speak multiple languages (Uschold 2001)
  - Different ontology languages are based on different paradigms (description logic, first order logic, frame-based, . . .)
  - It is not always possible to translate between languages. E.g. (Hayes, 96) shows that representations of time as points and as intervals are incompatible.

Agent communication - Ontology (2)

- Solutions:
  - Ontology negotiation (Bailin/Truszkowski, 2001)
  - Abstract Ontology Representation (Willmott, Constantinescu, Calisti, 2001)
  - Generic formalization of ontologies (Ontolingua)
  - "Big" ontologies (http://ontotext.com)

Conclusions — Information agents

- In order to build generic information agents there are unsolved problems on all levels of communication:
  - Transportation: KQML is not sufficient to interconnect agents on TCP/IP based networks.
  - Little experience with content languages use between alien agents.
  - Lack of common ontologies.
  - Little experience in dealing with malicious agents

Contact

http://www.uni-koblenz.de/ bthomas/MIA_HTML/
http://www.uni-koblenz.de/ag-ki/
gb@uni-koblenz.de