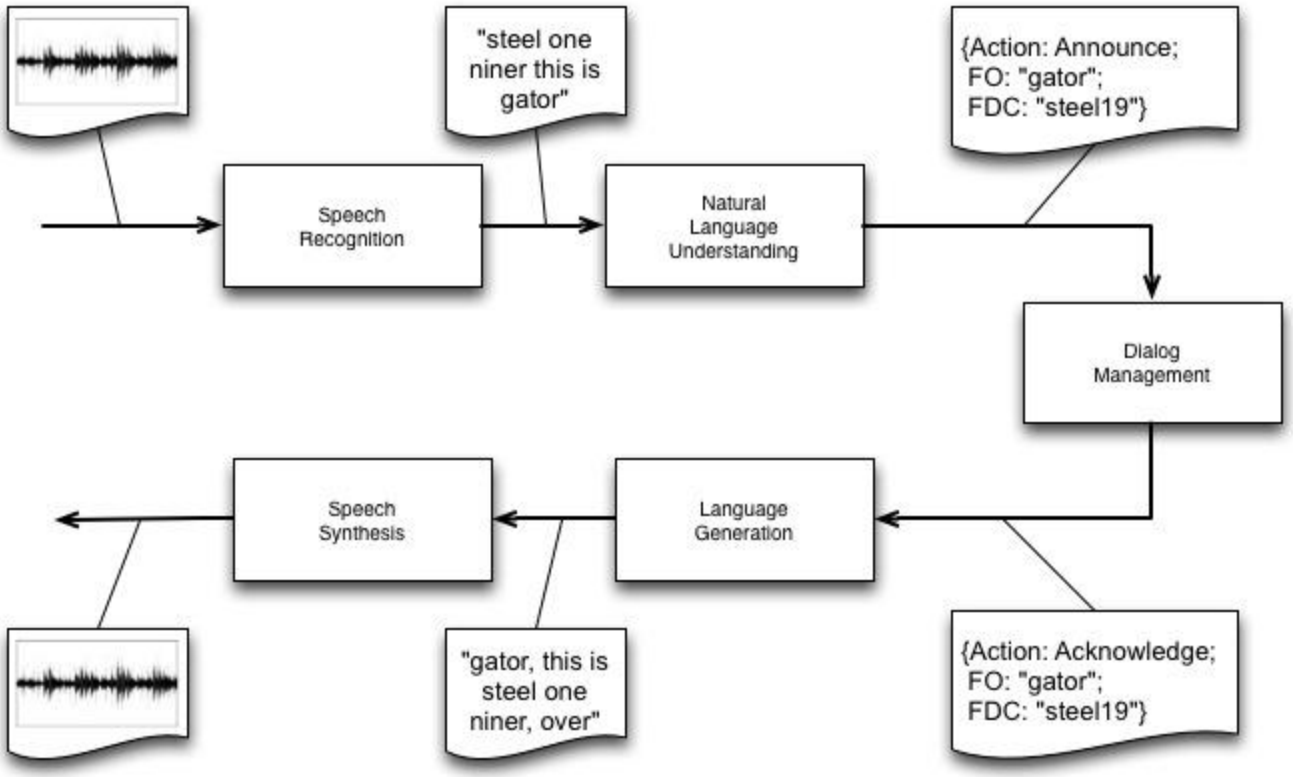
Investigation of the Information State Approach to Dialog Management using the DIPPER DME

ErikAnthony Harté
Ling 575 - Spring 2013

Project goals **(original)**

- Implement a Call For Fire (CFF) Dialog System
- Experiment with Information State Update and Dialog Move Engines

SDS Pipeline



DIPPER

- Developed by the Language Technology Group at The University of Edinburgh and CSLI Stanford
- Toolkit for prototyping dialog systems
- Modelled closely after TrindiKit, but...
  - Simpler framework
  - Not so Prolog-y
  - Tightly coupled with OAA
- Dipper_java
- Information State Update (ISU) approach
  - Information State - captures state of dialog, beliefs, goals, etc.
  - Update Rules - manages transitions between finite states

DIPPER - Information State

```
infostate(record([is:record([
                              grammar:atomic,
                              contact:atomic,
                              input:queue(move),
                              lastmoves:record([
                                                string:stack(word),
                                                act:stack(atomic),
                                                udr:stack(udr),
                                                conf:stack(atomic),
                                                int:list(interpretation)
                                               ]),
                              int:record([
                                          model:stack(model),
                                          drs:stack(drs)
                                         ])])])).
```

DIPPER - Update Rules

```
  urule(initialisation,
        [ is^grammar = '',
          empty(is^input) ],
        [ assign(is^contact,no),
          assign(is^listening,no),
          assign(is^grammar,'.Hotword') ]).

  urule(start_eavesdropping,
        [ is^contact = no,
          is^grammar = '.Hotword',
          is^listening = no,
          empty(is^input) ],
        [ solve(recognize(is^grammar,30,[Move|_]),[enqueue(is^input,Move),
assign(is^listening,no)]),
          assign(is^listening,yes) ]).
```

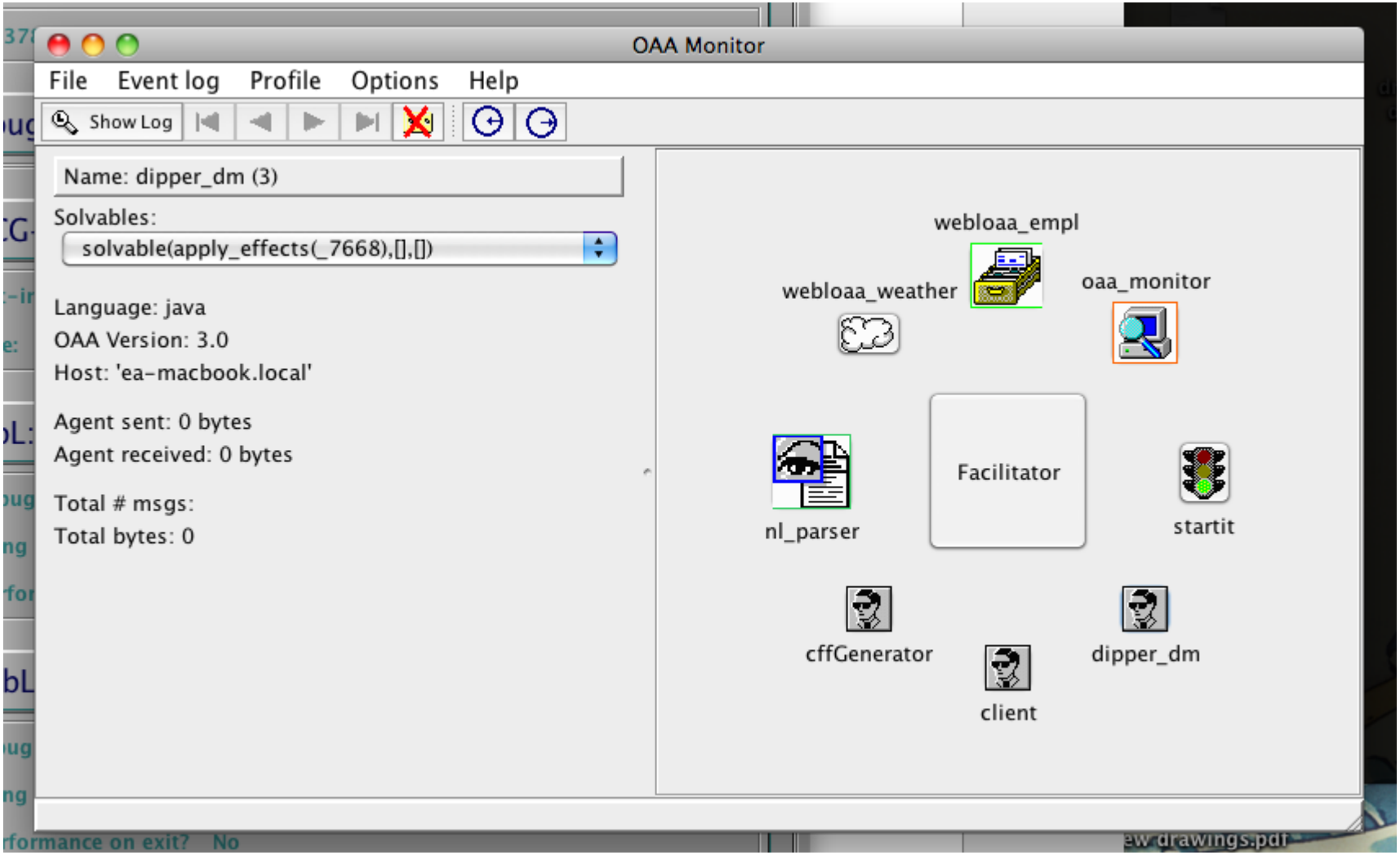
OAA - Open Agent Architecture

- Developed by SRI
- Framework for building communities of heterogenous software agents
- ICL - InterAgentCommunication Language (Prolog based)
- Agents communicate with each other through a Facilitator via ***solvables***
- Solvables announce and request agent capabilities within the community

Example: A mail service agent might register these solvables:

```
[solvable(send(mail, ToPerson, Msg), [callback(send_mail)], []),
  solvable(last_message(MessageNum),
           [type(data), single_value(true)],
           [write(true)]),
  solvable(get_message(MessageNum, Msg), [callback(get_mail)], [])
 ]).
```

OAA Monitor


OAA Monitor
File
Event log
Profile
Options
Help
Show Log
Name: dipper_dm (3)
Solvables:
solvable(apply_effects(_7668),[],[])
Language: java
OAA Version: 3.0
Host: 'ea-macbook.local'
Agent sent: 0 bytes
Agent received: 0 bytes
Total # msgs:
Total bytes: 0
webloaa_empl
webloaa_weather
oaa_monitor
Facilitator
nl_parser
startit
cffGenerator
client
dipper_dm

The Project: Implement CFF Dialog

Dialog Move Engine

- Information state
- Update rules

OAA Agents

- Natural Language Understanding
- Language Generation

Limited CFF Dialog

3 Phases

- Indentification
- Targeting
- Authentication

If time allows...

- Grounding
- Correction

Example:

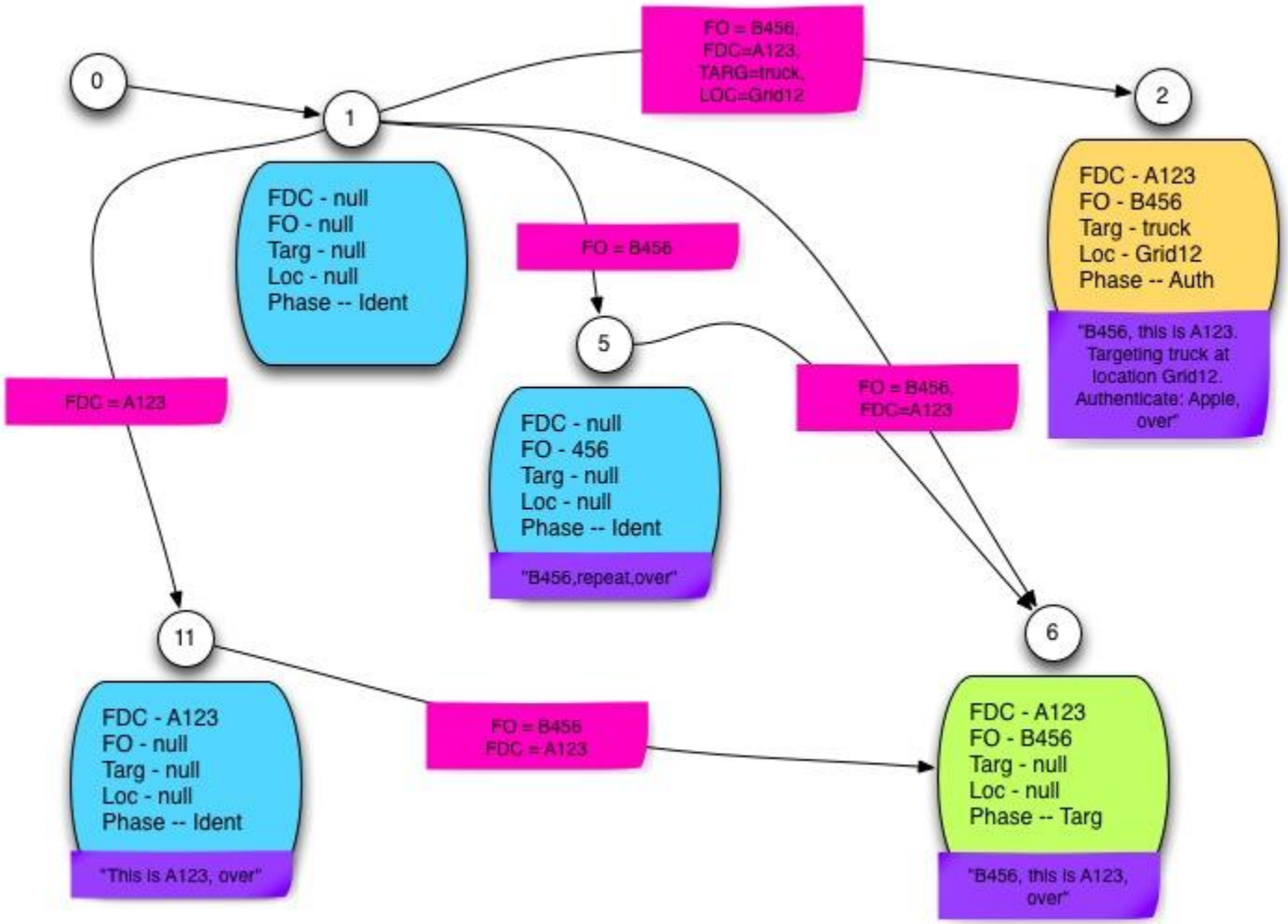
U: "B456, this is A123, over."

S: "A123, this is B456, out."

U: "Target is truck at location Grid12, over."

S: "Roger. Target truck at location Grid12, out."

Finite State Machine (~20 states)



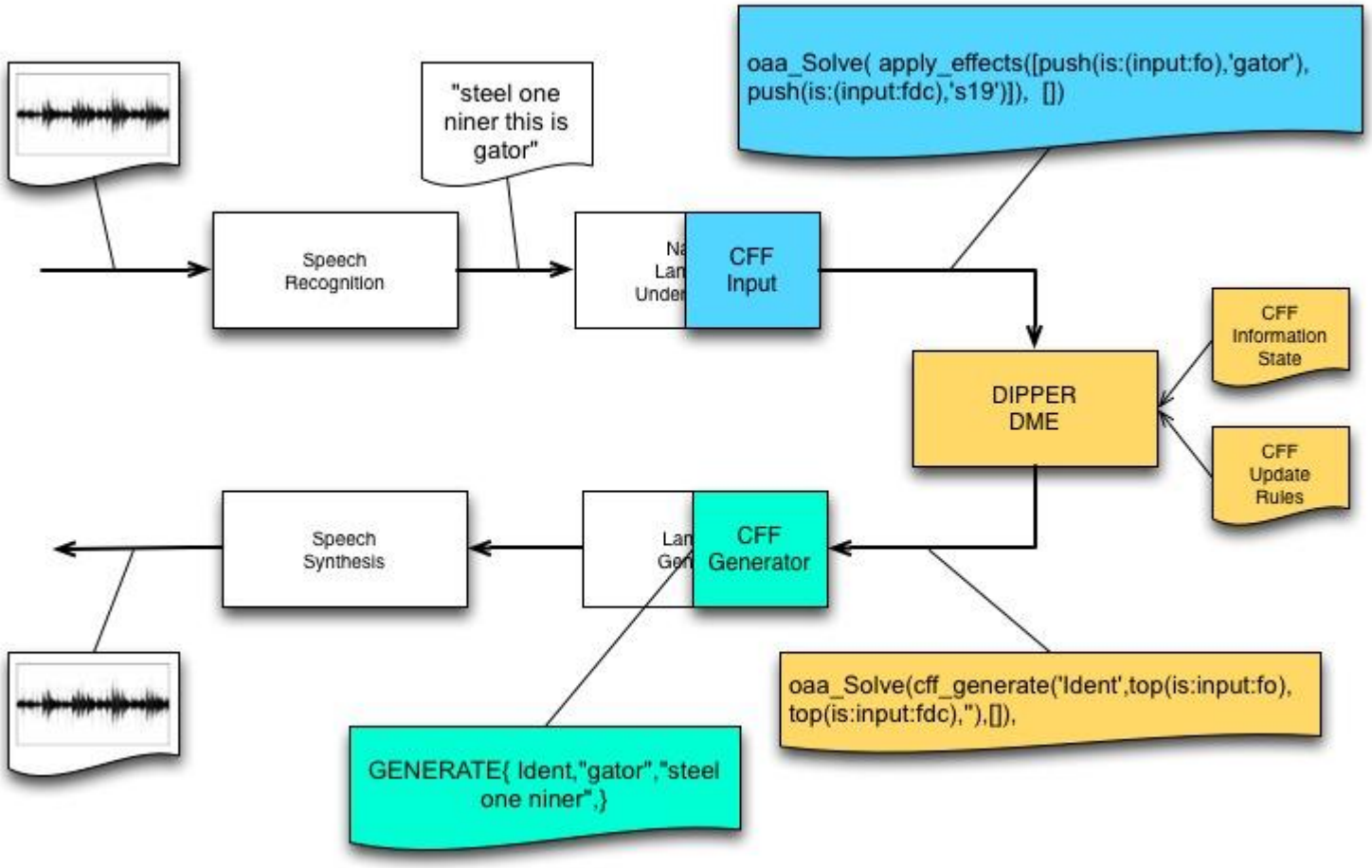
CFF Information State

```
infostate(record([is:record([
          phase:atomic,
        listening:atomic,

          input:record([
            fo:stack(atomic),
            fdc:stack(atomic),
            target:stack(atomic),
            location:stack(atomic)]),

          satisfied:record([
            fo:atomic,
            fdc:atomic,
            target:atomic,
            location:atomic]),

          auth:record([
           call:atomic,
           response:atomic,
           answer:atomic])

        ])])).
```

CFF Update Rules (~25 rules)

```
urule(rule4,
    [
     eq(is:phase,identification),
     eq(is:listening,yes),
     non_empty(is:input:fo),
     empty(is:input:fdc),
     empty(is:input:target),
     empty(is:input:location),
     eq(is:satisfied:fo,''),
     eq(is:satisfied:fdc,''),
     eq(is:satisfied:target,''),
     eq(is:satisfied:location,'')
    ],
    [
     assign(is:listening,no),
     solve(cff_generate('RequestRepeat',top(is:input:fo), ",''),[]),
     pop(is:input:fo),
     assign(is:listening,yes)
    ]
   ).
```

Implementation



Conclusions

- ISU interesting and very flexible - supports a wide variety of dialog management approaches
- OAA architecture allows for independent modules to be built
- Toolkits allow relatively easy prototyping, and (with effort) the ability to build usable systems

Issues

- Difficult to use, especially TrindiKit
- Prolog knowledge essential
- Little documentation, particularly DIPPER. As well as lack of current support.
- Difficult to manage rules and state with complicated dialogs

References

1. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture (2003). Johan Bos , Ewan Klein , Oliver Lemon , Tetsushi Oka. In 4th SIGdial Workshop on Discourse and Dialogue

2. Staffan Larsson and David Traum (2000): Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. In Natural Language Engineering Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering, Cambridge University Press, U.K. (pp. 323-340, 18 pages)

3. David Traum and Staffan Larsson (2003): The Information State Approach to Dialogue Management. To appear in Smith and Kuppevelt (eds.): Current and New Directions in Discourse & Dialogue, Kluwer Academic Publishers. (pp. 325-353, 28 pages)

4. Bohus, Dan & Alexander I. Rudnicky (2009), "The RavenClaw dialog management framework: Architecture and systems", Computer Speech & Language