CS 599: Computational Models of Dialogue Modelling: Fall 2005 Lecture 4: Frame-based and Information-state based Approaches

David Traum

traum@ict.usc.edu

http://www.ict.usc.edu/~traum

Outline

- Frame-based approach
  - Example systems: MIT
- Frame+agenda
  - CMU
- Information-state approach
  - Trindikit
  - Other kits
- Example information-based theories & systems
  - EDIS

Transaction Dialogues

- User has a request
- System needs info from user to process request
- Dialogue proceeds as:
  - User specifies request
  - System gathers necessary info
    - Q&A
    - Spontaenous assertion from user
  - System looks up information & provides response

Frame-based Approach

- Also called form-based (MIT)
- Central data structure is frame with slots
  - DM is monitoring frame, filling in slots
- Used for transaction dialogues
- Generalizes finite-state approach by allowing multiple paths to acquire info
- Frame:
  - Set of information needed
  - Context for utterance interpretation
  - Context for dialogue progress
- Allows mixed initiative

Example: MIT Wheels system

- Domain: searching used car ads
- Transaction domain + constraint satisfaction
- No slots are mandatory,
  - try to find the best set of matches
  - Try to find an appropriate # of matches

Example: MIT Jupiter System (1)

- Retrieval of weather forecast domain
  - Multiple sources
  - Content processing
  - Information on demand
  - Context
- 1-888-573-8255

MIT Jupiter System (2)

- Uses Galaxy architecture
  - SUMMIT ASR
    - 2000 word vocabuluary, 1-9% OOV
  - TINA NL understanding
    - Creates semantic frames from text
    - Used for both query understanding (user)
    - Content understanding (web-based weather text)
  - GENESIS generation
    - User text
    - SQL queries
    - Keyword-value
  - Dialogue control table
    - Conditions for operations
    - context

Problems with Frames

- Not easily applicable to complex tasks
  - May not be a single frame
  - Dynamic construction of information
  - User access to “product”

Agenda + Frame (CMU Communicator)

- Product:
  - hierarchical composition of frames
- Process:
  - Agenda
    - Generalization of stack
    - Ordered list of topics
    - List of handlers

Example: CMU Communicator System

The Information State Approach to Dialogue Modelling:

Some Results of the TRINDI Project

David Traum

USC Institute for Creative Technologies

traum@ict.usc.edu

TRINDI Project

- Task-Oriented Instructional Dialogue
- European Union Telematics, 2yr project (1998-2000)
- ~15 Researchers
- Consortium: U Gothenburg, U Edinburgh, U Saarlandes, SRI Cambridge, Xerox

Motivating Problems

- Dialogue theories are largely incomparable
  - despite often similar intended coverage
  - e.g., motivation for answering questions:
    - cooperativity vs. obligations vs. QUD structure
  - Heterogeneous building blocks
- Large gap between dialogue models in systems and broad-coverage theories
- Dialogue systems are hard to build
  - despite rapid progress in ASR, TTS, NLP
  - hard to convert systems to new domains
  - insufficient attention to `theoretical' concerns

Deficiencies of Previous Dialogue Theories

- Inappropriate for direct implementation
  - Some aspects too vague
    - e.g., Relevance Theory (a la Sperber and Wilson)
  - some aspects too complex for efficient computation
    - e.g., Implicit Belief using Modal Predicate Logic
- Hard to evaluate/compare with other theories
    - even when covering same dialogue phenomena
    - Heterogeneous building blocks
  - How to combine, e.g., mentalistic and structural

Deficiencies of Current Dialogue Systems

- Software engineering challenge
  - combining heterogeneous sub-systems
- Domain/Task specific design
  - little carried over to next system
- Insufficient attention to dialogue structure
  - Dialogue usually conceived as FSM
    - inflexible interaction
    - does not scale to large tasks

Partial Solution: Dialogue Toolkits

- Software Integration
  (OAA,Trains/Trips,Verbmobil)
- FSM Dialogue Kits (Nuance, OGI, …)
- Slot-Filling (Phillips)
- Current Development Kits:
  - Utterance-based (DARPA Communicator)
  - ⇒ Information-based (TrindiKit)

Approach to Problems

- Information State approach to formalizing theories of dialogue modelling
- Dialogue Move Engine (TrindiKit) for implementing a dialogue modelling theory
- Example implementations
- Comparative experimentation, enhancements, & evaluation

Information State Theories of Dialogue

- Statics
  - **Informational components** (functional spec)
    - e.g., QUD, common ground, dialogue history, ...
  - **formal representations** (acessibility)
    - e.g., lists, records, DRSes, …
- Dynamics
  - **dialogue moves**
    - abstractions of i/o (e.g., speech acts)
  - **update rules** - atomic updates
  - **update strategy** - coordinated application of rules

Sample GoDiS information state

- PRIVATE =
  - AGENDA = { **findout(?return)** }
  - PLAN =
    - **findout(?λx.month(x))**
    - **findout(?λx.class(x))**
    - **respond(?λx.price(x))**
  - BEL = { }
  - TMP = (*same as SHARED*)
- SHARED =
  - COM =
    - **dest(paris)**
    - **transport(plane)**
    - **task(get_price_info)**
  - QUD = < **λx.origin(x)** >
  - LM = { **ask(sys, λx.origin(x))** }

Sample GoDiS update rule

- integrateAnswer

pre:
- in(SHARED.LM, answer(usr, A))
- fst(SHARED.QUD, Q)
- relevant_answer(Q, A)

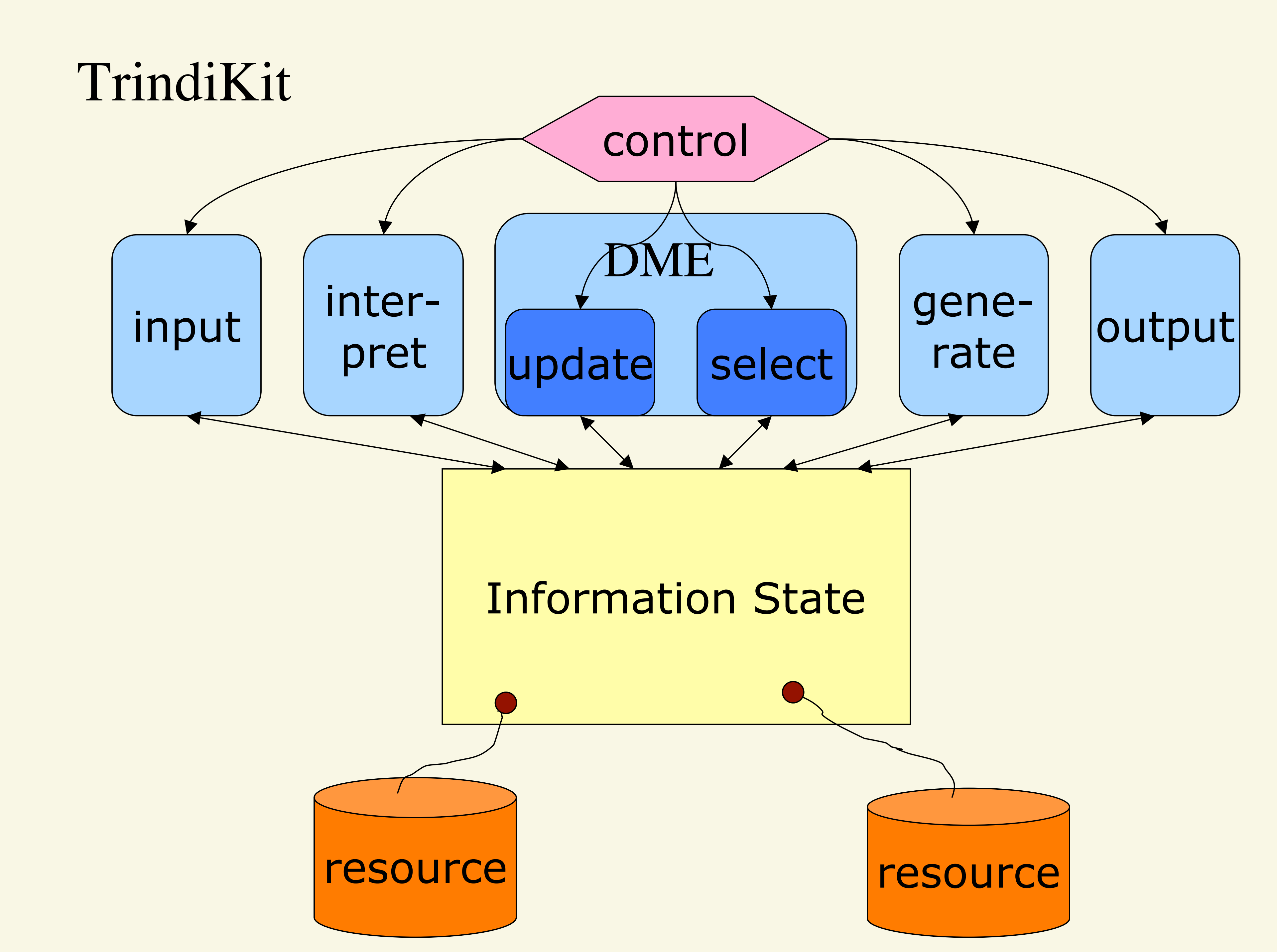
eff:
- pop(SHARED.QUD)
- reduce(Q, A, P)
- add(SHARED.COM, P)

Dialogue Move Engine

- Handles Dialogue Management tasks:
  - consumes observed dialogue moves
  - updates information state
  - produces new dialogue moves to be performed
- Can be implemented as:
  - Update (&Selection) Rules
  - Update Algorithm

TrindiKit

- Architecture based on information states
- Modules (dialogue move engine, input, interpretation, generation, output etc.) access the information state
- Resources (databases, lexicons, domain knowledge etc.)


TrindiKit
control
input
inter-
pret
DME
update
select
gene-
rate
output
Information State
resource
resource

TrindiKit Features

- Explicit information state data-structure
  - makes systems more transparent
  - closer to dialogue processing theory
  - easier comparison of theories
- modularity for simple and efficient reconfiguration and reusability
- rapid prototyping

TrindiKiT Includes

- A library of datatype definitions
  - conditions and operations
- facilities for writing update rules and algorithms
- tools for visualizing information state
- debugging facilities
- A library of basic ready-made modules for i/o, interpretation, generation, etc.
- Resource interfaces

Building a TrindiKit system

Build or select from existing components:

- Type of information state (DRS, record, ...)
- A set of dialogue moves
- Information state update rules,
- DME Module algorithm(s), including control algorithm
- Resources: databases, grammars, plan libraries etc., or external modules

Building a system

domain knowledge (resources) → domain-specific system

dialogue theory (IS, rules, moves etc) → domain-independent DME

software engineering (basic types, control flow) → TRINDIKIT

TrindiKit Systems

- GoDiS (Larsson et al) – information state: Questions Under Discussion
- MIDAS – DRS information state, first-order reasoning (Bos &Gabsdil, 2000)
- EDIS – PTT Information State, (Matheson et al 2000)
- SRI Autoroute – information state based on Conversational Game Theory (Lewin 2000)

  Robust Interpretation (Milward 1999)

System Comparisons

- Cross-IS Theories: SRI vs. EDIS on AutoRoute Dialogues
- Different formalizations: PTT using DRSes or Records
- Different Update strategies:
  - GoDiS with or without plan accomodation
  - Midas using different grounding strategies
- Different Languages, Tasks, and interactivity
  - GoDiS: English vs. Swedish
  - GoDiS: AutoRoute vs. Travel Agent
  - IMDIS: dialogue vs. text

Potential Impact

- Better development environment for formal dialogue theories
  - easy testing/revision of theories
  - comparison across theories
- Closer integration of theories and systems
- Better dialogue system development
  - Information state vs. dialogue state
  - extension to other domains

Post-Trindi Applications

- Siridus Project (EU 2000-)
  - Command and negotiative dialogues
  - Spanish
  - GoDiS, SRI
- IBL for Mobile Robots (U Edinburgh)
  - Midas
- Tutoring Electricity (U Edinburgh)
  - EDIS

Successor Toolkits

- TrindiKit revisions
- Dipper
- Midiki

EDIS SYSTEM

- Uses PTT theory
- Trindikit implementation
- Autoroute domain

PTT Informational Components

- Separate Views for System and User
  (System assumptions about User)
- Private, Public, and Semi-public components of View captures *grounding* process (Clark& Schaefer '87)
  - GND represents common ground
  - set of DUs represent partitioned semi-public information introduced but not (yet) grounded
  - UDUs structure accessible ungrounded DUs
- (Semi-)Public Information includes:
  - public events
  - social commitments of participants
- Private Information includes
  - Intentions
  - Beliefs

EDIS Formalization of Information Components

- Record (AVM) for Views, with fields for each dialogue participant:
  - GND: PT-Rec
    Public Information
  - UDUS: list of accessible DU IDs
  - CDU (DU-ID,PT-Rec)
    current Discourse Unit
  - PDU (DU-ID,PT-Rec)
    penultimate Discourse Unit
  - INT: list of intended actions

- PT-REC contains:
  - DH: list of dialogue acts
    Dialogue History of performed dialogue acts
  - OBL: list of action types
    Obligations of participants to perform actions
  - SCP: list of states
    Social Commitments of agents to Propositions
  - COND: list of implications
    relevant conditional anticipated effects

PTT Information State

$$\begin{bmatrix} \mathrm{G} & : & \mathit{PT\text{-}R} \\ \mathrm{CDU} & : & \begin{bmatrix} \mathrm{C} & : & \mathit{PT\text{-}R} \\ \mathrm{ID} & : & \mathrm{DU\text{-}ID} \end{bmatrix} \\ \mathrm{PDU} & : & \begin{bmatrix} \mathrm{C} & : & \mathit{PT\text{-}R} \\ \mathrm{ID} & : & \mathrm{DU\text{-}ID} \end{bmatrix} \\ \mathrm{UDUs} & : & \mathrm{List(DU\text{-}ID)} \\ \mathrm{INT} & : & \mathrm{List(Action)} \end{bmatrix}$$

$$\mathit{PT\text{-}R} : \begin{bmatrix} \mathrm{DH} & : & \mathrm{List(Action)} \\ \mathrm{OBL} & : & \mathrm{List(Action)} \\ \mathrm{SCP} & : & \mathrm{List(Prop)} \\ \mathrm{COND} & : & \mathrm{List(Action)} \end{bmatrix}$$

EDIS Dialogue Moves

- Forward-looking
  - assert(dp,Prop)
  - check(dp, Prop)
  - direct (dp,act-type)
  - info-request(dp,Q)
- Backward Looking
  - Address(dp,act)
    - accept
    - agree
    - answer
  - Understanding Act
    - Acknowledge(dp,DU-ID)

Update Strategy

- Deliberation (produce new intentions)
- Acting on intentions (produce output dialogue moves)
- Update based on an observed utterance
  1. Create a new DU and push it on top of UDUs.
  2. Perform updates for backwards grounding acts.
  3. For other types, record in cdu.dh and apply the update rules for act class
  4. Apply inference update rules to all parts of the IS which contain newly added acts.

Update Rules

- effects of observed dialogue acts
  - formalized in terms of social commitments
- inference
  - Obligation Resolution
  - Conditional Resolution
  - Intention Resolution
- Deliberation
  - adopting new intentions

Dialogue Act Effect Updates

act	ID:2, **ack**(DP,DU1)
effect	peRec(w.Gnd,w.pdu.tognd)
effect	remove(DU1,UDUS)
act	ID:c, **forward-looking-act**(DP)
effect	push(obl,**u-act**(**o**(DP),CDU.id))
act	ID:2, **accept**(DP,ID2)
effect	*accomplished via rule resolution*
act	ID:2, **agree**(DP,ID2)
effect	push(scp,**scp**(DP,**P**(ID2)))
act	ID:2, **answer**(DP,ID2,ID3)
effect	push(scp,**ans**(DP,**Q**(ID2),**P**(ID2)))
act	ID:2, **assert**(DP,PROP)
effect	push(scp,**scp**(DP,PROP))
effect	push(cond,**accept**(**o**(DP),ID)→ **scp**(**o**(DP),PROP))
act	ID:1, **assert**(DP,PROP)
effect	push(cond,**accept**(**o**(DP),ID)→ **scp**(**o**(DP),PROP))
act	ID:2, **check**(DP,PROP)
effect	push(obl,**address**(**o**(DP),ID))
effect	push(cond,**agree**(**o**(DP),ID) → **scp**(DP,PROP))
act	ID:2, **direct**(DP,Act)
effect	push(obl,**address**(**o**(DP),ID))
effect	push(cond,**accept**(**o**(DP),ID) → **obl**(**o**(DP),Act))
act	ID:2, **info_request**(DP,Q)
effect	push(obl,**address**(**o**(DP),ID))

Deliberation Factors

- obligations
  - to perform understanding acts
  - to address previous dialogue acts
  - to perform other actions
- potential obligations
  that would result if another act were performed,
  as represented in the cond field (or CDU.OBL)
- insufficiently understood dialogue acts
  with a 1 confidence level in cdu.dh
- intentions to perform complex acts

Deliberation Rules

1. Grounding:
   OBL U-act, everything in CDU understood
   $\Rightarrow$ ack(W,CDU)

2. Address:
   OBL address act
   $\Rightarrow$ accept, agree, or answer

3. Anticipatory Planning:
   INT act1 $\wedge$ COND act1 $\rightarrow$ OBL act2
   $\Rightarrow$ act2 add an intention to perform an action

4. SubGoal: Int(act1) $\wedge$ NextSubact(Act1,Act2)
   $\Rightarrow$ Act2

   (a) check CDU.DH:1

   (b) info-request

Sample Autoroute Dialogue

W **WIZARD**	**CALLER**
[1]: How can I help you?	[2]: A route please
[3]: Where would you like to start?	[4]: Malvern
[5]: Great Malvern?	[6]: Yes
[7]: Where do you want to go?	[8]: Edwinstowe
[9]: Edwinstowe in Nottingham?	[10]: Yes
[11]: When do you want to leave?	[12]: Six pm
[13]: Leaving at 6 p.m.?	[14]: Yes
[15]: Do you want the quickest or the shortest route?	[16]: Quickest
[17]: Please wait while your route is calculated.	

InfoState after [2]: A route please

- **W:**
  - **GND:**
    - OBL: ⟨ **understandingAct**(W,DU3), **address**(C,CA2) ⟩
    - DH: ⟨ CA3: C2, **acknowledge**(C,DU2), CA2: C2, **info_request**(W,?helpform) ⟩
    - SCP: < >
    - COND: < >
  - **UDUS:** <DU3>
  - **PDU:**
    - TOGND:
      - OBL: <**address**(C,CA2)>
      - DH: <CA2: C2, **info_request**(W,?helpform)>
      - SCP: < >
      - COND: < >
    - ID: DU2
  - **CDU:**
    - TOGND:
      - OBL: <**address**(W,CA6)>
      - DH: ⟨ CA6: C2, **direct**(C,giveroute(W)), CA5: C2, **answer**(C,CA2,CA4), CA4: C2, **assert**(C,want(C,route)) ⟩
      - SCP: <**scp**(C,want(C,route))>
      - COND: <**accept**(W,CA6) **->** **obl**(W,giveroute(W))>
    - ID: DU3
  - **INT:** ⟨ **info_request**(W,?start), **giveroute**(W), **accept**(W,CA6), **acknowledge**(W,DU3) ⟩
- **C:** [INT: <**getroute**(C)>]

InfoState after [4]: Malvern, prompting check

- W:
  - GND:
    - OBL: ⟨ **giveroute**(W), **understandingAct**(W,DU5), **address**(C,CA8) ⟩
    - DH: ⟨ CA10: C2, **acknowledge**(C,DU4), CA9: C2, **accept**(W,CA6), CA8: C2, **info_request**(W,?start) ⟩
    - SCP: < >
    - COND: < >
  - UDUS: <DU5>
  - PDU:
    - TOGND:
      - OBL: <**address**(C,CA8)>
      - DH: ⟨ CA9: C2, **accept**(W,CA6), CA8: C2, **info_request**(W,?start) ⟩
      - SCP: < >
      - COND: < >
    - ID: DU4
  - CDU:
    - TOGND:
      - OBL: < >
      - DH: ⟨ CA12: C2, **answer**(C,CA8,CA11), CA11: C1, **assert**(C,start(malvern)) ⟩
      - SCP: < >
      - COND: < >
    - ID: DU5
  - INT: ⟨ **check**(W,start(malvern)), **acknowledge**(W,DU5), **giveroute**(W) ⟩
- C: [INT: <**getroute**(C)>]

InfoState after [5]: Great Malvern?

```
W:  GND:   OBL:   ⟨ understandingAct( C,DU6 )
                    giveroute( W ) ⟩
           DH:    ⟨ CA13: C2, acknowledge( W,DU5 )
                    CA12: C2, answer( C,CA8 )
                    CA11: C1, assert( C,start(malvern) ) ⟩
           SCP:   < >
           COND:  < >
    UDUS:  <DU6>
    PDU:   TOGND: OBL:   < >
                  DH:    ⟨ CA12: C2, answer( C,CA8,CA11 )
                           CA11: C1, assert( C,start(malvern) ) ⟩
                  SCP:   < >
                  COND:  < >
           ID:    DU5
    CDU:   TOGND: OBL:   <address( C,CA14 )>
                  DH:    <CA14: C2, check( W,start(malvern) )>
                  SCP:   < >
                  COND:  <agree( C,CA14 ) -> scp( W,start(malvern) )>
           ID:    DU6
    INT:   <giveroute( W )>
C:  [INT: <getroute( C )>]
```

InfoState after [7]: Where do you want to go?

- W:
  - GND:
    - OBL: ⟨ **understandingAct**(C,DU8), **giveroute**(W) ⟩
    - DH: ⟨ CA17: C2, **acknowledge**(W,DU7), CA16: C2, **agree**(C,CA14) ⟩
    - SCP: ⟨ **scp**(C,**start**(malvern)), **scp**(W,**start**(malvern)) ⟩
    - COND: < >
  - UDUS: <DU8>
  - PDU:
    - TOGND:
      - OBL: < >
      - DH: <CA16: C2, **agree**(C,CA14)>
      - SCP: <**scp**(C,**start**(malvern))>
      - COND: < >
    - ID: DU7
  - CDU:
    - TOGND:
      - OBL: <**address**(C,CA18)>
      - DH: <CA18: C2, **info_request**(W,?dest)>
      - SCP: < >
      - COND: < >
    - ID: DU8
  - INT: <**giveroute**(W)>
- C: [INT: <**getroute**(C)>]

- W:
  - GND:
    - OBL: ⟨ **understandingAct**(W,DU3), **address**(C,CA2) ⟩
    - DH: ⟨ CA3: C2, **acknowledge**(C,DU2), CA2: C2, **info_request**(W,?helpform) ⟩
    - SCP: < >
    - COND: < >
  - UDUS: <DU3>
  - PDU:
    - TOGND:
      - OBL: <**address**(C,CA2)>
      - DH: <CA2: C2, **info_request**(W,?helpform)>
      - SCP: < >
      - COND: < >
    - ID: DU2
  - CDU:
    - TOGND:
      - OBL: <**address**(W,CA6)>
      - DH: ⟨ CA6: C2, **direct**(C,giveroute(W)), CA5: C2, **answer**(C,CA2,CA4), CA4: C2, **assert**(C,want(C,route)) ⟩
      - SCP: <**scp**(C,want(C,route))>
      - COND: <**accept**(W,CA6) **->** **obl**(W,giveroute(W))>
    - ID: DU3
  - INT: ⟨ **info_request**(W,?start), **giveroute**(W), **accept**(W,CA6), **acknowledge**(W,DU3) ⟩
- C: [INT: <**getroute**(C)>]