Towards a Foundation for Comprehensive Scheme Support in Argumentative Dialogue Games

Simon Wells
University of Aberdeen

CMNA 13 @ ICAIL 2013, Rome

Intro: Argument & Dialogue

- "Tackling the conceptual leap from problem domain to deployment of arg-oriented tools (within that domain)"
- Mixed human/agent arena supporting some combination of:
  - Computational Efficiency
  - Scrutability & Introspection
  - Alignment with Human Reasoning & Interaction Processes

What's the story?

- Schemes look like a good way to structure a KB about a problem domain
- Chaining schemes - good way to structure interaction within the domain - even richer if using dialogue games
- Looks to be a good alignment between arguments (schemes) & dialogue (games) [*NB*. Atkinson, Reed]

Games & Schemes

- Explore (& optionally extend) the (scheme structured) KB
- Feed arg process
  - e.g. determine what to say next or which areas of domain exploration have been neglected
- Ultimately: interact more easily with intelligent computational systems using mechanisms that are more human-oriented

But it's not that easy...

- Practically it is not so straightforward to align games & schemes
- Generally, games do not do argument very well
  - For "well" read: 'explicitly' in terms of mapping locutions to argument building/manipulating acts
  - all feels a bit rudimentary...

Expressiveness

- Currently, often treat locutional content as atomic: ranging from statement variables, & propositions, through quite complex sentences of language *L*
- Interpretation is often left to the listener
- Makes computational game players more difficult to produce
- Essentially a form of loosely interpreted DuckTyping - not always clear what the status of a piece of content is (argument, premise, conclusion, ???)

Game Description

- Schemes are quite simple & reasonable well structured
- Games can be complex
  - How do we structure a game so that it usefully supports schemes & scheme components?
  - How do we describe this support so it is useful/transparent (meta-level)?
  - Can we provide useful guidelines to game designers?

Assumptions

- Computationally accessible (automated & unambiguous) scheme sets (with namespacing)
  - e.g. can retrieve {scheme_name, conclusion_desc, minor/major_premise_desc, [critical_questions]}

Annotating Locutional Content

- Remove ambiguity from interpretation of what speaker meant
- If I say that P is my conclusion & I want it to be interpreted as such then why not be able to label it as such?
- e.g. assert("conclusion": "p")
- rather than the current fashion for overloading locutions:
- e.g. assert_conclusion("p")

Similarly

- We could go further & explicitly link moves & content with specific schemes:
  - e.g. assert ("conclusion": "p", "scheme":"expert_opinion")
- or (to extend the earlier example):
  - e.g. assert_scheme_conclusion("p")

Describing Games

- When describing a game:
  - {optionally | mandatorily} label content as some | all {argument_part | scheme_part} using key:value style notation
- When playing a game:
  - Dependent upon rules; {optionally | mandatorily} label content as some | all {argument_part | scheme_part} using key:value style notation

Conclusions

- Unpacking a lot that has been previously assembled into the locution label, or left to interpretation, or deus ex machina solutions
  - e.g. "assume some mechanism that can recognise that an argument conforming to a particular scheme has been uttered"
- (If) Games generate more explicit data - (naively assume) improves computational tractability
  - Remove class of interpretation related problems that depended upon working out: where the content stood in the argument, what the arg meant, etc.

Future Work/Questions

- How does this affect strategy?
  - Removing ambiguity could remove a whole class of rhetorical devices
  - can't rely on misrepresenting the form of an opponent's argument
- How does this affect interpretation & analysis?