Towards a Declarative Approach to Constructing Dialogue Games

Mark Snaith[1], Simon Wells[2]

[1]Robert Gordon University, Garthdee House, Garthdee Road, Aberdeen, AB10 7QB, Scotland, UK
[2]Edinburgh Napier University, Merchiston Campus, Edinburgh, EH10 5DT, Scotland, UK

Abstract

In this paper we sketch a new approach to the development of dialogue games that builds upon the knowledge gained from several decades of dialogue game research across a variety of communities and which leverages the capabilities of the Dialogue Game Description Language as a means to describe the constituent parts of dialogue games. Our ultimate aim is to produce a method for rapidly describing and implementing games that conform to the designer's needs by declaring what is required and then automatically constructing the game from components, called 'fragments', that are distilled from existing dialogue games.

1. Introduction & Motivation

Dialogue games are generally designed to fulfil a specific purpose, for example, modelling a specific type of dialogue such as the deliberation dialogue game of McBurney et al [1], or investigating specific contexts of argumentation or dialogue as illustrated in the sequence of elaborations of game rules for prohibiting the fallacy of *Petitio Principii* as found in the dialogue of papers between Woods & Walton [2, 3, 4] on the one hand and of Mackenzie [5, 6] on the other. A marked feature of this approach was that once the wider set of performatives was chosen, much of the investigation focused upon small changes to individual constituent rules. A similar approach has happened over the intervening years as various challenges in dialogue modelling for agent communication, legal argument, educational interaction, and policy formulation have been investigated. This has gone beyond specific types of dialogue as envisaged by Walton and Krabbe [7] to encompass specific areas of human activity, whether educational dialogue games such as Ravenscroft's 'CTG' [8] and Yuan's 'DE' [9] or the coaching games of Snaith *et al* [10]. In each context a specific problem is identified, and a dialogue game that codifies the interaction dynamics of the situation is produced, often in combination with wider theoretical contributions. This is not to belittle these very valuable contributions that have shed light on many aspects of dialogical interaction. Rather we propose that this exact progress has put us in the position of being able to target a wide range of characteristic dialogue types, identify games for each of those types, and locutions and rules within those games which pertain to

CMNA'21: Workshop on Computational Models of Natural Argument, September 02, 2021, Online
m.snaith@rgu.ac.uk (M. Snaith); s.wells@napier.ac.uk (S. Wells)
https://www3.rgu.ac.uk/dmstaff/snaith-mark (M. Snaith); https://www.simonwells.org (S. Wells)
0000-0002-0877-7063 (M. Snaith); 0000-0003-4512-7868 (S. Wells)

a broad range of dialogical contexts of interaction. A standard way to design and develop dialogue games has involved the selection of an appropriate set of locutions, often from an existing and well understood game, then specialising those locutions to the new context. DC [5] has been a popular basic framework for such research and has yielded specialised variations such as Yuan's aforementioned game 'DE'. Additionally there have been two other threads of dialogue game research that are important and relate variously to testing games and to describing them. Firstly the investigation of desiderata [11] and 'Drosophila' [12] for evaluating and testing dialogue games according to various criteria and standards. Secondly, the design of description languages for producing executable specifications of dialogue games and the development of game agnostic dialogue run-times for executing game specifications. Together these approaches enable us to describe, in a consistent and verifiable way, the rules for a specific game, because of the description language, and then to evaluate that game, either by how well the game satisfies a given set of criteria (identified through appropriate desiderata), or by how well the game performs in a given scenario (identified by an appropriate 'Drosophila'), where criteria and scenario both match the design goals for the game. Existing games provide a useful set of starting rules, locution sets, and permissible or prohibited behaviours, work on desiderata provides useful benchmarks for evaluation, and work on description languages provides a method of unambiguous description and execution. This paper lays the foundations for an approach to dialogue game design and construction that builds upon insights from all of this previous work and seeks to both learn from and integrate these approaches whilst also imposing a more user friendly declarative approach to game construction. Two key aspects of this approach are to make the construction of dialogue games declarative, and to reuse suitably abstracted fragments of existing games as templates, to scaffold the generation of new games rapidly, accurately, and on demand by non-experts - for example, to underpin interactions in systems for conversational AI [13]. The remainder of this paper is organised as follows: Section 2 discusses related work, Section 3 describes how DGDL can be re-purposed as the foundation for a dialogue rule fragment library and introduces elements of a game design and construction methodology, Section 4 illustrates the proposed process using an existing game from a coaching scenario, and finally Section 5 reflects upon this approach and describes some directions for future work.

2. Background & Related Work

A related approach is the Lightweight Coordination Calculus (LCC) due to Robertson [14]. Whilst this approach shares some of our ideals in generating flexible and (semi-)automated communication protocols, it is a low level approach that is well suited to software agent coordination, but omits the focus on norm-oriented human communication that characterises much of dialogue game research, favouring an agent centred role that attempts to bridge between computational process communication and multi-agent institutions. A second early relevant approach was the effort to do run-time negotiation of the semantics of agent communication protocols [15]. This made an important contribution to the concept of protocol as a constructed object within a shared space as well as the idea that communication protocol is fungible. Finally a more recent approach to designing and implementing dialogue games can be seen in

ProtOCL [16] which exploits the unified modelling language (UML) and the Object Constraint Language (OCL), technologies that are more commonly used to design and implement Object Oriented software systems. In ProtOCL, dialogue games are specified using a series of annotated diagrams which describe facets of the desired dialogue system. Code, in the form of a Java Object hierarchy is generated and incorporated into a wider software system. Both the LCC and the negotiating semantics approaches will likely form a basis for further work that exploits our proposed fragment library, especially in extending the system towards automated use by agents, however for the moment we are focusing upon dialogue game development as an essentially human-centric design activity akin to the ProtOCL approach.

3. Dialogue Fragments & the DGDL

This approach to dialogue description exploits the functionality of the Dialogue Game Description Language (DGDL) [17]. In this section we give a high level overview of the DGDL before focusing upon game descriptions expressed in the DGDL before finally investigating how such descriptions can be taken apart to form "dialogue fragments".

3.1. DGDL

The DGDL is a domain specific language for describing the rules of dialogues games in a format that can be executed on a dialogue management run-time (such as DGEP [18] or Adamant [19]). This way a single language can be used to describe a flexible and diverse multiplicity of games which can then be selected at run-time (rather than compile or design time). This avoids the need to create a novel implementation each time a new game is devised, but also provides a route towards implementation for non-programmers. The DGDL is founded upon an extended Backus-Naur Form (EBNF) grammar [20] that enables games to be described in a way that is expressive, accounting for a wide variety of rule types, consistent, so that game descriptions are cohesive, and syntactically verifiable, so that each description can be relied upon to be correctly expressed within the bounds of the DGDL. The DGDL essentially enables the participants of the game to be identified, the turn structure to be specified, the stores of artefacts generated during the dialogue to be designated, and for the rules and interactions to be formulated. Note that rules and interactions are very similar, they define a set of requirements that must hold and a consequent set of effects to apply but they differ only in how they are applied. Interactions describe moves that are explicitly made by a participant during their turn whereas rules are applied on either a turn-wise or a move-wise basis irrespective of the interactions associated with a given turn or move. The distinction can be seen in the check for a termination condition, regardless of the move made, after each turn a check is made on whether a given win-loss condition holds.

3.2. DGDL game descriptions

We'll not reproduce the entire grammar here, for reasons of space, but a minimal valid game description for a game named '*Simple*' from [21] is illustrated in figure 1. In *Simple* there are two players, labelled 'Player1' and 'Player2', who each possess a public artefact store that is

```
Simple{
   {turns,magnitude:single,ordering:strict}
   {players,min:2,max:2}
   {player,id:Player1}
   {player,id:Player2}
   {store,id:CStore,owner:Player1,structure:set,visibility:public}
   {store,id:CStore,owner:Player2,structure:set,visibility:public}
   {Assert,{p},``I assert that'',{store(add, {p}, CStore, Speaker)}}
}
```

Figure 1: An example two player dialogue game called 'Simple' described in the DGDL.

structured as a set of artefacts. Each turn the players may make a single move with a strictly alternating turn structure. There is a single move available to the players, the assert move, which enters the asserted statement into the artefact store of the speaker. Most dialogue games are more complex than this. Whilst the specification of players and turns is generally similar, most games define a rich set of interactions for the players to make, rather than the single 'Assert' move in the example, and a diverse set of effects, in terms of game artefacts, to apply once those moves have been made. Note however that there is a conceptual and practical difference between the DGDL grammar and a DGDL description. The grammar defines an abstract space of possible games whilst a DGDL description defines a single, specific, concrete game that exists within that space. It is the interval between abstract grammar and concrete game rule that we seek to exploit in this work.

3.3. Deconstructing DGDL descriptions into fragments

Rather than an entire game description we desire fragments of a game that encapsulate an entire design concept from the source game and which are suitable for reconstructing into entirely new games. This introduces a notion of portability of rules between DGDL described games. If a given game implements a particular set of rules, which would be useful to include in another game, in order to make the same behaviour available in that other game, then fragments are one element of a means to achieve this. A basic fragment is a valid DGDL expression, starting from any valid left-hand-side (LHS) grammar rule, rather than merely the start rule, and completing it through to it's terminal values. Whereas a 'DGDL game description' describes an entire game, a fragment describes only a part of a game. So all fragments are syntactically valid sub-expressions of the DGDL grammar. Because such basic fragments are particular to specific games, in order to make them more flexible, elements of a fragment can be replaced with their equivalents from the DGDL grammar originating from the same LHS starting point to yield a semi-instantiated 'template fragment'. So, for example, where a fragment might refer, in it's concrete instance, to a specific effect, e.g. "{store(add, {p}, CStore, Speaker)}" this could be replaced with the "RuleBody" grammar declaration to indicate that all other aspects of the fragment should remain the same but that different rule bodies could be applied, for example, a different artefact store effect. The idea is to provide an intermediate step between the two extremes of the highly abstract and incomplete grammar level rules and the completely concrete game description expressions. These intermediate states are our abstract fragments

One or more fragments may be collected together to define a dialogical behaviour context. We desire to be able to take a declaration such as "prohibit circular reasoning" and then to generate games which fulfil this criterion on the basis of existing DGDL fragments. Note that this will be extremely difficult to completely achieve automatically and without error but provides a route towards more flexible, on the fly, dialogue protocol generation that accounts directly for the discoveries and achievements in dialogue game research over past decades. Fragments from multiple different dialogue games can be brought together to form a single library. The intuition is that games which have been decomposed into fragments can subsequently be recombined to form new games based upon the desired characteristics of the new game. A fragment library will comprise a database of fragments from existing dialogue games together with an interface for assembling fragments into new games.

3.4. Elements of a Process for Constructing Dialogue Games from Fragments

Assuming a library of game fragments, suitably represented as templates for further specialisation in specific generated games, let's now consider the process of developing a new game from a more declarative perspective. The goal is to be able to describe what we want our dialogue game to do in terms of the characteristics of the resulting game. We propose that much of this process would be performed in a specialised game development UI whose input was a description of the desired game and whose output is a valid and fully-formed DGDL document. A variety of user interfaces could be envisaged from a GUI that enables the user to describe their needs using standard GUI widgets (selections, drop-downs, text inputs, and whatnot). Various forms of text-based UI could also be envisaged, including a traditional application programming interface (API) or, fittingly, a conversational interface, perhaps in the spirit of the recently announced GitHub co-pilot tool. A starting place is to define a minimal, archetypal set of locutions that conform to each supported dialogue type and which scaffold the construction of the remainder of the game. We name such a set of locutions the base game for dialogue type *n*. The base game is a starting place, and the addition of rules to it leads to a specific derived game. If more than one dialogue type is selected then there are two further organisational models that can account for how the games work together. The first uses a Walton style shift model to move between dialogue instances, and the second uses a compound model in which the locutions for each dialogue type are amalgamated into a single whole. There are benefits to each approach and both are supported by the DGDL. Once a base game has been selected, the user is instructed to select desirable properties for their game. For example, prohibiting, prescribing, or permitting specific behaviours. Each supported property will correspond to either an existing fragment, or else will require a new rule to be defined (in which case the new rule becomes a fragment in the library for subsequent reuse. At this point a DGDL description can be generated. In order to determine whether the resulting game is sound automated testing of the game is performed applying a mixture of desiderata/properties measurement, Drosophila/scenario based testing, and a form of arguing agent competition [22]. After testing, the game can be imported to a DGDL run-time such as the Dialogue Game Execution Platform (DGEP) [18] or A DiAlogue MANagement Tool (ADAMANT) [19].

LR1	$C \in \chi$ can justify a goal (*Justify*), revise a goal (*Revise*), challenge a goal (*Challenge*), or accept a goal (*accept*): 1. $Justify(g, r)$ when they justify the goal g with reason r 2. $Revise(g, g')$ when they revise the goal g to new goal g' 3. $Challenge(g)$ when they challenge a goal g proposed by another coach, or the patient 4. $Accept(g)$ when they accept a goal g proposed by a patient
LR2	*LC*, in addition to those locutions available to all coaches, can propose a goal (*Propose*): 1. $Propose(g)$ when they propose the goal g
LR3	*P* can accept a goal (*Accept*) and be unsure about a goal (*Unsure*): 1. $Accept(g)$ when they accept a goal g 2. $Unsure(g)$ when they are unsure about a goal g 3. $Revise(g, g')$ when they revise the goal g to a new goal g'

Table 1
GSDG Locution rules

4. Worked Example

Here, we motivate our proposed Dialogue Fragment Library by showing how an existing dialogue game can be broken down into fragments of certain dialogue types. Our intention with this example is to demonstrate that 1) a domain-specific dialogue game consists of more general sets of locutions and rules; and 2) the game itself could have originally been constructed from such fragments. For the sake of space, we will not provide the DGDL fragments themselves, but will instead leave their specifications implicit. We use the goal-setting dialogue game (*GSDG*) specified in [10]. Goal-setting is a common technique used in behaviour change, where one or more coaches (experts) engage with a patient to help them towards an achievable goal (e.g. a number of daily exercise minutes) [23]. A dialogue based on *GSDG* is between at least two participants: a "lead coach" (*LC*) with expertise in the area in which the goal is being set, e.g. physical activity, and a patient (*P*). All other (optional) participants are "coaches" ($C_1 \ldots C_n$) with expertise differing from that of *LC* (e.g. diet, cognitive etc.). The locution and structural rules of *GSDG*, shown in Tables 1 and 2 respectively, contain elements of persuasion and negotiation. A dialogue begins with *LC* proposing a goal, which *P* can either *accept* or be *unsure* about. Acceptance by *P* terminates the dialogue, whereas expressing uncertainty continues the process, which includes *LC* being able to justify the goal (persuasion), or *P* and *LC* repeatedly providing counter-proposals until a mutually-agreeable goal is found (negotiation). A final element is the ability for another coach (*C*) to themselves query the goal proposed by *LC* if *P* has expressed uncertainty. We now demonstrate how *GSDG* can be deconstructed into smaller dialogue fragments.

4.1. Simple persuasion game

The basic structure of *GSDG* is a simple asymmetric persuasion game between only the lead coach and the patient. If we were to generalise this persuasion game, with P_1 ("player 1") taking the place of lead coach and P_2 taking the place of the patient, we have the following structure (where the steps are followed in sequence, unless stated):

SR1	All players can perform only one move per turn
SR2	*LC* moves first with $Propose(g)$
SR3	After *LC* performs $Propose(g)$, *P* can perform: (1) $accept(g)$, or (2) $Unsure(g)$
SR4	After *P* performs $Unsure(g)$, $C \in \chi \setminus \{C_1\}$, where C_1 is the (possibly Lead) Coach to whom *P* responded, can perform: (1) $Challenge(g)$; or C_1 can perform (1) $Justify(g, r)$
SR5	After $C \in \chi \setminus \{C_1\}$ performs $Challenge(g)$, where C_1 is the (possibly Lead) Coach to whom the challenge is aimed, C_1 can perform: (1) $Justify(g, r)$, or (2) $Revise(g, g')$
SR6	After C_1 performs $Justify(g, r)$, $C \in \chi \setminus \{C_1\}$ can perform: (1) $Challenge(r)$, or (2) $Revise(g, g')$; or *P* can perform: (1) $Accept(g)$, or (2) $Revise(g, g')$
SR7	After $C \in \chi$ performs $Revise(g, g')$, *P* can perform: (1) $Accept(g')$, or (2) $Unsure(g')$; or $C_1 \in \chi \setminus \{C\}$ can perform: (1) $Challenge(g')$
SR8	After *P* performs $Revise(g, g')$, $C \in \chi$ can perform: (1) $Accept(g')$, or (2) $Revise(g', g'')$

Table 2
GSDG Structural rules

1. P_1: $Propose(g)$
2. P_2: $Unsure(g) \rightarrow$ step 3
   $Accept(g) \rightarrow terminates$
3. P_1: $Justify(g, r)$
4. P_2: $Accept(g) \rightarrow terminates$

This game is somewhat trivial, but is nonetheless valid. Notionally, it models a situation where P_2 can either outright accept a proposal from P_1, or demand justification. If a justification is forthcoming, P_2 is obliged to accept the original proposal. We henceforth refer to this simple protocol as *SPD* (Simple Persuasion Dialogue).

4.2. Adding elements of negotiation

A key principle of goal-setting is that the patient must take ownership of the goal so they are motivated to achieve it [23]. This is modelled in *GSDG* by incorporating elements of negotiation, allowing the patient to propose their own goal via a "*Revise*" locution, and subsequently allowing the lead coach to propose a further alternative goal. These exchanges can be isolated from *GSDG* and generalised thus:

1. P_2: $Revise(g, g') \rightarrow$ step 2
   $Accept(g) \rightarrow terminates$
2. P_1: $Revise(g', g'')$
   $Accept(g') \rightarrow terminates$

While this doesn't represent a complete dialogue game, we do nevertheless name the functionality it provides as *SNE* (Simple Negotiation Exchange). Using *SPD* as our starting point, *SNE* is incorporated in two places: for P_2, *Revise* is available as an alternative to *Accept* following a *Justify*; for P_1, *Revise* and *Accept* are then available. P_1 using a *Revise* move effectively triggers a new dialogue with g'' as the new topic.

1. P_1: $Propose(g)$

2. P_2: $Unsure(g) \rightarrow$ step 3
   $Accept(g) \rightarrow$ *terminates*
3. P_1: $Justify(g, r)$
4. P_2: $\mathbf{Revise(g, g')} \rightarrow$ step 5
   $Accept(g) \rightarrow$ *terminates*
5. P_1: $\mathbf{Revise(g', g'')} \rightarrow$ step 2
   $Accept(g') \rightarrow$ *terminates*

This game, *SPD-SNE*, can be described as follows:

- A game based on *SPD*
- Following a $Justify$ move from P_1, P_2 can begin an *SNE*
- If the *SNE* does not lead to termination, proceed to step 2 (effectively starting a new *SPD-SNE* albeit without the initial *Propose* move)

4.3. Third-party challenging

The final aspect of *GSDG* is another coach being able to challenge the lead coach's proposed goal. This challenge differs slightly from that available to the patient (via an *Unsure* move) in that the *Challenge* move is available to the other coach if and only if the patient themselves expresses uncertainty (via an *Unsure* move). This is intended to model the behaviour observed in [10] where a second coach can challenge *LC*'s proposed goal as a way of helping the patient reflect on whether they wish to propose an alternative. Adding third-party challenging is relatively trivial insofar as its use does not mandate any new locution types, but it does have an impact on the overall structure of possible dialogues. Following a *Challenge* move, P_1 is permitted to immediately revise the challenged goal, effectively triggering a new dialogue with the revised goal as the topic:

1. P_1: $Propose(g)$
2. P_2: $Unsure(g) \rightarrow$ step 3 or 4
   $Accept(g) \rightarrow$ *terminates*
3. $\mathbf{P_3}$: **Challenge(g)**
4. P_1: $Justify(g, r) \rightarrow$ step 5
   $\mathbf{Revise(g, g')}$ (if P_3 moved *Challenge*) $\rightarrow$ step 2
5. P_2: $Revise(g, g') \rightarrow$ step 6
   $Accept(g) \rightarrow$ *terminates*
6. P_1: $Revise(g', g'') \rightarrow$ step 2
   $Accept(g') \rightarrow$ *terminates*

This modified version of *SPD-SNE* can be described as follows:

- A game based on *SPD-SNE* with an additional participant P_3
- Following an *Unsure* move from P_2, P_3 can perform a *Challenge* move, after which P_1 can $Justify$ or *Revise*, and the dialogue continues per the existing rules of *SPD-SNE*

4.4. Summary

The aim of this worked example has been to show that an existing domain-specific dialogue game, *GSDG* can be deconstructed into fragments consisting of more general dialogical interactions and that, in principle, the game could have been constructed from those fragments in the first place. Underpinning *GSDG* are three distinct elements: a simple persuasion game (*SPD*), a simple negotiation exchange (*SNE*), and third-party challenging. Individually, none of these are intrinsically linked to goal-setting; it is in combining them we arrive at a dialogue game that models the theoretical behaviour of goal-setting.

5. Discussion & Future Work

We've sketched out how the DGDL can form the central scaffold for a declarative approach to game development that builds upon and exploits much previous work. Key questions remain regarding the degree to which we've explored the possible space of dialogue games, how well the DGDL supports complete exploration of that space, and how many rules and behaviours have been encoded into DGDL expressions. It remains unclear to what degree existing rules can be recombined automatically, but we are committed to the idea that we should be learning from, and building upon, the lessons from existing dialogues games. It's also unclear to what degree automated testing of new games can be sound and exhaustive. However, for the first time, it appears that the key pieces to enable an advance in dialogue game design, construction, testing, and deployment are in place, potentially enabling wider exploitation of this approach to dialogue research within the wider conversational AI agenda,.

References

[1] P. McBurney, D. Hitchcock, S. Parsons, The eightfold way of deliberation dialogue, International Journal of Intelligent Systems (2007) 95–132.

[2] J. Woods, D. N. Walton, Petito principii, Synthese 31 (1977) 107–128.

[3] J. Woods, D. N. Walton, Arresting circles in formal dialogues, Journal Of Philosophical Logic 7 (1978) 73–90.

[4] J. Woods, D. N. Walton, Question-begging and cumulativeness in dialectical games, Nous 16 (1982) 585–605.

[5] J. D. Mackenzie, Question begging in non-cumulative systems, Journal Of Philosophical Logic 8 (1979) 117–133.

[6] J. D. Mackenzie, Begging the question in dialogue, Australasian Journal Of Philosophy 62 (1984) 174–181.

[7] D. N. Walton, E. C. W. Krabbe, Commitment in Dialogue, SUNY series in Logic and Language, State University of New York Press, 1995.

[8] A. Ravenscroft, Promoting thinking and conceptual change with digital dialogue games, Computer Assisted Learning 23 (2007) 453–465.

[9] T. Yuan, Human Computer Debate, A Computational Dialectics Approach, Ph.D. thesis, Leeds Metropolitan University, 2004.

[10] M. Snaith, D. de Franco, T. Beinema, H. op den Akker, A. Pease, A dialogue game for multi-party goal-seting in health coaching, in: Proceedings of the 7th International Conference on Computational Models of Argument (COMMA 2018), Warsaw, Poland, 2018, pp. 337–344.
[11] P. McBurney, S. Parsons, M. Wooldridge, Desiderata for agent argumentation protocols, Proceedings of the First AAMAS (2002) 402–409.
[12] S. Wells, C. Reed, A drosophila for computational dialectics, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2005), 2005.
[13] T. Beinema, D. Davison, D. Reidsma, O. Banos, M. Bruijnes, B. Donval, Álvaro Fides Valero, D. Heylen, D. Hofs, G. Huizing, R. B. Kantharaju, R. Klaassen, J. Kolkmeier, K. Konsolakis, A. Pease, C. Pelachaud, D. Simonetti, M. Snaith, V. Traver, J. van Loon, J. Visser, M. Weusthof, F. Yunus, H. Hermens, , H. op den Akker, Agents united: An open platform for multi-agent conversational systems, in: Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents (ACM IVA 2020), 2021. To appear.
[14] D. Robertson, Lightweight coordination calculus for agent systems: Retrospective and prospective, in: Declarative Agent Languages and Technologies IX. DALT 2011., volume 7169 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2012.
[15] C. Reed, T. Norman, N. Jennings, Negotiating the semantics of agent communication languages, Computational Intelligence 18 (2002) 229–252.
[16] T. Yuan, S. Wells, Protocl: Specifying dialogue games using uml and ocl, in: F. Grasso, N. Green, C. Reed (Eds.), Thirteenth International Workshop on Computational Models of Natural Argument (CMNA13), 2013, pp. 74–85.
[17] S. Wells, C. Reed, A domain specific language for describing diverse systems of dialogue, Journal of Applied Logic 10 (2012) 309–329.
[18] J. Lawrence, M. Snaith, B. Konat, K. Budzynska, C. Reed, Debating technology for dialogical argument: Sensemaking, engagement and analytics, ACM Transactions on Internet Technology 17 (2017) 24:1–24:23.
[19] S. Wells, The open argumentation platform (OAPL), in: Proceedings of Computational Models of Argument. (COMMA 2020), Frontiers in Artifical Intelligence, IOS Press, 2020, pp. 465–476.
[20] N. Wirth, What can we do about the unnecessary diversity of notation for syntactic definitions?, Communications of the ACM 20 (1977) 822–823.
[21] S. Wells, Supporting argumentation schemes in argumentative dialogue games, Studies in Logic, Grammar and Rhetoric (SLGR) 36 (2014) 171–191.
[22] S. Wells, C. Reed, Towards an arguing agents competition: Architectural considerations, in: Proceedings of the 8th International Workshop on Computational Models of Natural Argument (CMNA8), 2008.
[23] E. A. Locke, G. P. Latham, Building a practically useful theory of goal setting and task motivation: A 35-year odyssey., American psychologist 57 (2002) 705.