Computational Models of Argument
S. Parsons et al. (Eds.)
IOS Press, 2014

doi:10.3233/978-1-61499-436-7-59

Lakatos Games for Mathematical Argument

Alison PEASE [a], Katarzyna BUDZYNSKA [a,b], John LAWRENCE [a], and Chris REED [a]
[a] *School of Computing, University of Dundee, UK*
[b] *Institute of Philosophy and Sociology, Polish Academy of Sciences, Poland*

Abstract. We present a dialogue game representation of Lakatos's theory of justification and discovery in mathematics and describe our implementation of the game. As well as demonstrating a new domain for dialogue games, this could provide the basis for systems which can be used to aid mathematicians in their work.

Keywords. Lakatos, dialogue games, mathematical argument, mathematical conversations.

1. Introduction

The study of mathematical practice emphasises social aspects of mathematical reasoning, and in particular the role that dialectical negotiation plays in the communal development and understanding of concepts, conjectures and proofs (for example, see [10,4]). Lakatos [12] was the forerunner of these ideas: he held a fallibilist picture of mathematics in which mathematical reasoning is defeasible, and was interested in studying proof-as-process, that is, the social processes of proof construction, as opposed to proof-as-product, that is, the 'final' polished products. It is these same informal aspects of mathematical reasoning that we are concerned with in this paper: we hold that work in argumentation theory can be applied to informal, defeasible aspects of mathematical reasoning; conversely, the fertile domain of mathematical reasoning can be used to evaluate and extend general argumentation structures.

Lakatos outlined various methods by which proof-as-process develops, and emphasised the dialectical aspect in the style of his book, which takes the form of a dialogue in a classroom. In the following paper we describe our representation of Lakatosian methods, or patterns of dialogue, in terms of a formal dialogue game, and our implementation of the game.

The relationship between the study of mathematical practice and argumentation theory is not well-explored, despite deep connections. There are, however, some examples of bridges between the two fields. While Toulmin initially developed his layout to describe non-mathematical argument, he did consider some mathematical arguments, for example, [19, pp.135-136], and he later applied the layout to Theaetetus's proof that there are exactly five platonic solids [20]. Since then Aberdein has shown that Toulmin's argumentation structure can represent more complex mathematical proofs [3], and Alcolea [5] applied the structure to meta-level mathematical argument. Krabbe [11] also considers the relationship between proofs and arguments, exploring the application of

argumentation-theoretic concepts to mathematical proofs, and presenting a dialectical and rhetorical view of proofs. Aberdein [4] furthermore elaborates the dialogical context of mathematical argument, in terms of Walton's six types of dialogue, and proposes types of proof dialogue in these terms. In [16] Pease *et al.* identify connections between standard argumentation structures and Lakatos's methods and describe a computational representation of Lakatos's theory. In [14] Pease and Aberdein explore connections and overlapping ideas between Lakatos's theory and theories in argumentation.

2. Lakatos's patterns of dialogue

Lakatos demonstrated his argument by presenting case studies of the development of Euler's conjecture that for any polyhedron, the number of vertices (V) minus the number of edges (E) plus the number of faces (F) is equal to two, and Cauchy's proof of the conjecture that the limit of any convergent series of continuous functions is itself continuous. We outline Lakatos's methods below: crudely speaking, monster-barring is concerned with concept development, exception-barring with conjecture development, and the method of proofs and refutations with proof development. However, these are not independent processes; much of Lakatos's work stressed the interdependence of these three aspects of theory formation.

The method of **surrender** consists of abandoning a conjecture in light of a counterexample.

Piecemeal exclusion deals with exceptions by excluding a whole class of counterexamples. This is done by generalising from a counterexample to a class of counterexamples which have certain properties. For instance, the students generalise from the hollow cube to *polyhedra with cavities*, and then modify Euler's conjecture to 'for any polyhedron without cavities, $V - E + F = 2$'.

Strategic withdrawal uses positive examples of a conjecture and generalises from these to a class of object, and then limits the domain of the conjecture to this class. For instance, the students generalise from regular polyhedra to *convex polyhedra*, and then modify Euler's conjecture to 'for any convex polyhedron, $V - E + F = 2$'.

Monster-barring/monster-adjusting is a way of excluding an unwanted counterexample. This method starts with the argument that a 'counterexample' can be ignored because it is *not* a counterexample, as it is not within the claimed concept definition. Rather, the object is seen as a monster which should not be allowed to disrupt a harmonious theorem. For instance, one of the students suggests that the hollow cube (a cube with a cube-shaped hole in it) is a counterexample to Euler's conjecture, since $V - E + F = 16 - 24 + 12 = 4$. Another student uses monster-barring to argue that the hollow cube does not threaten the conjecture as it is not in fact a polyhedron. The concept polyhedron then becomes the focus of the discussion: using this method, the original conjecture is unchanged, but the meaning of the terms in it may change. Monster-adjusting is similar, in that one reinterprets an object in such a way that it is no longer a counterexample, although in this case the object is still seen as belonging to the domain of the conjecture.

Lemma incorporation works by distinguishing global and local counterexamples. The former is one which is a counterexample to the main conjecture, and the latter is a counterexample to one of the proof steps (or lemmas). A counterexample may be both global and local, or one and not the other. When faced with a counterexample, the first step is to determine which type it is. If it is both global and local, *i.e.*, there is a problem both with the argument and the conclusion, then one should modify the conjecture by incorporating the problematic proof step as a condition. If it is local but not global, *i.e.*, the conclusion may still be correct but the reasons for believing it are flawed, then one should modify the problematic proof step but leave the conjecture unchanged. If it is global but not local, *i.e.*, there is a problem with the conclusion but no obvious flaw in the reasoning which led to the conclusion, then one should look for a hidden assumption in the proof step, then modify the proof and the conjecture by making the assumption an explicit condition.

Proofs and refutations consists of using the proof steps to suggest counterexamples (by looking for objects which would violate them). For any counterexamples found, it is determined whether they are local or global counterexamples, and then lemma incorporation is performed.

In Lakatos's case study of Euler's conjecture the conjecture and each lemma in the proof are implications, and the structure of the proof is a hypothetical syllogism. While Lakatos expected that his theory apply to other types of conjecture and other forms of proof[1], we adopt this structure in our formalisation and subsequent implementation.

3. Dialogue protocol for collaborative mathematics

In this section we make the first steps towards specifying a formal system for playing a game of collaborative mathematics. Here, we define only a protocol while selected information about other types of rules (such as commitment and transformation rules) are only mentioned in the informal description of the protocol in the second part of this section.

A protocol for the Lakatos Game (LG) system describes the possible exchanges of locutions between a proponent of a proof, P, and an opponent, O. Using lemmas *l, k,* ..., the player P is trying to prove that the conjecture c holds. Let also m,n be a counterexample (a monster) to the conjecture, and r – a class of objects that are positive or negative examples of c. Lakatos games are an example of persuasion dialogues (see [17] for an overview). A game consists of attacks, for O, and defences, for P, of the conjecture c. The rules **P4.3** and **P10.2** correspond to accepting conjecture c, and **P5.8** and **P11.2** to rejecting it. In other words, they terminate a collaborative proof.

Referring to the protocol below, P creates a proof in the following way. He proposes a conjecture, say c_1 (**P1**), which is added to the Conjecture store. Then, the proponent introduces a lemma, say l (**P2**), which is added to the Lemma store. P can then add another lemma or complete the proof. If he gives another lemma this loops adding lemmas k, ... (**P3.1**). When the proponent stops giving lemmas and wants to complete the proof,

[1] [12] was published posthumously and Lakatos never considered it to be finished.

he can execute it by saying *ProofDone* for: either already introduced conjectures, i.e. all of them that are in the Conjecture store, e.g. c_1 (see **P3.2**), or for a new conjecture (**P3.3**), say c_2, that he needs to introduce, e.g. via the method of piecemeal exclusion.

P1 A player *P* moves first with *Conjecture(c)*; then each player contribute a locution according to the rules below

P2 After *Conjecture(c)*, *P* introduces a lemma *Lemma(l)*

P3 After *Lemma(l)*, *P* can:

1. perform a sequence of locutions introducing more lemmas: *Lemma(k)*, ...
2. end the proof of existing conjectures *x* by saying: *ProofDone(x)*
3. end the proof of a new conjecture *c* by saying: *ProofDone(c)*

P4 After *ProofDone(c)*:

1. *O* can reply *Counter(m, c)*
2. *P* can reply *StrategicWithdrawaldr(r)*
3. *O* can reply *Accept*

P5 After *Counter(m, c)*, *P* can reply:

1. *PiecemealExclusion(r)*
2. *StrategicWithdrawal(r)*
3. *MonsterBar(m)*
4. *MonsterAdjust(m)*
5. *GlobalLemmaInc(l, m)* for *l*
6. *LocalLemmaInc(l, m)* for *l*
7. *HybridLemmaInc(l, m)* for *l*
8. *Surrender*

P6 After *PiecemealExclusion(r)*, *P* replies: *ProofDone(c)*

P7 After *StrategicWithdrawal(r)*, *P* replies: *ProofDone(c)*

P8 After *MonsterBar(m)*, *O* can reply:

1. *MonsterAccept(m)*
2. *MonsterReject(m)*

P9 After *MonsterAdjust(m)*, *O* can reply:

1. *MonsterAccept(m)*
2. *MonsterReject(m)*

P10 After *MonsterAccept(m)*, *O* can reply:

1. *Counter(n, x)* for a current conjecture *x*
2. *Accept*

P11 After *MonsterReject(m)*, *P* can reply:

1. *PiecemealExclusion(r)*
2. *Surrender*

P12 After *GlobalLemmaInc(l, m)*, *P* replies: *HiddenAssLemma(h)*

P13 After *HiddenAssLemma(h)*, *P* replies: *Lemma(k)*

P14 After *LocalLemmaInc(l, m)*, *P* replies: *FixLemma(k, m)*

P15 After *FixLemma(l, m)*, *P* replies: *PrevLemma(k, m)*

P16 After *PrevLemma(l, m)*, *P* replies: *Lemma(k)*

P17 After *HybridLemmaInc(l, m)*, *P* replies: *ProofDone(c)*

The conjecture has typically a form: *"For each x, if x has a property A, then it has a property B"* (lemmas have such a form too). After proposing the proof (see the rules **P1**–**P3**), the opponent can attack *c* by showing a counterexample, i.e. a monster *m* "there is an object *m* that is *A*, but not *B*" (**P4.1**). The proponent has several defences to choose from: (1) defences that modifies the conjecture (**P5.1**–**P5.2**); (2) defences that modifies the concept *A* or *B* (**P5.3**–**P5.4**); or (3) defences that modifies the lemmas (**P5.5**–**P5.7**).

If the proponent chooses option (1), he needs to introduce a new property into the conjecture. As a result, he "repairs" the proof and proposes a modified conjecture (**P6** and **P7**). For example, in the method of piecemeal exclusion, *P* reformulates the conjecture to a form *"For each x, if x has a property A but not D, then it has a property B"*. Next, the opponent can accept the new conjecture or start a new series of attacks.

If the proponent chooses option (2), his defence strategy is to show that *m* is not a counterexample by redefining property *A* or *B*. For example, in monster barring method, *P* claims that actually *m* does not have a property *A* (or more precisely – that *m* does not satisfy a new definition of *A*). In fact, after *MonsterBar(m)* and *MonsterAdjust(m)*

the players start another (embedded) game about the definition of the concept *A* or *B*. This dialogue terminates with *O* accepting new definition (**P8.1** and **P9.1**), or rejecting it (**P8.2** and **P9.2**). In the first case, the opponent can continue attacking by using another counterexample (another monster; **P10.1**), and in the second – *P* can continue defending by using piecemeal exclusion **P11.1**.

The key difference between the strategies (1) and (2) is that in the first one the definition of *A* does not change, but the proponent introduces a new property *A*' which contains those *x* that are *A* but not *D*. On the other hand, in the second case the conjecture itself does not change, but *A* obtains a new definition *def*'. In other words, the latter strategy has an impact on the rest of the theory, i.e., it changes the meaning of all other conjectures that include *A*.

If option (3) is chosen, *P* is manipulating lemmas by: (a) adding new, hidden lemma (**P12–P13**); (b) replacing faulty lemma (**P14–P16**); or (c) modifying the conjecture to incorporate problematic proof step (**P17**) (see also Sect. 4 for more details on these techniques).

4. Identifying the current proof

The Dialogue Game Execution Platform (DGEP) [21] interprets dialogue game specifications expressed in an amended version of the Dialogue Game Description Language (DGDL) and allows for dialogues to take place based on the rules of the specification whilst creating an Argument Interchange Format (AIF) [7] structure to capture the moves being made. One of the advantages of using DGEP is that it not only provides a robust platform for the execution of a dialogue protocol, but also creates argument structures as a side effect. Although these structures are designed to represent argument, their closest counterparts in formal logical systems are not formal theories or sets of propositions, but proofs. If the description of LG in DGDL is adequate, therefore, it should yield proofs in AIF.

There is, however, a challenge. The infrastructure of the Argument Web is founded upon several assumptions. One cornerstone is the assumption of monotonicity: once an argument has been committed to the Argument Web, it is there in perpetuity. Although individual arguers may change their position or retract their arguments, the arguments themselves must remain available. In general, monotonicity in the Argument Web [6] can be considered to be similar to the monotonicity that characterises academic literature as a whole. Even explicit retractions, which have become infamous lately, are nevertheless incremental and monotonic additions to the academic record.

However, if the Argument Web is to capture the emerging proofs as they are developed, extended, redefined and reframed by a Lakatos game, how can the intrinsic non-monotonicity be reflected by monotonic infrastructure? Our goal is to ensure that the semantics of the moves in the game, as they are defined in terms of AIF updates, and expressed in DGDL, should yield argument structures with a specific set of properties. These argument structures should be submittable to a simple, algorithmic procedure which will yield precisely the proof that represents the current shared understanding at any point in the game. To deliver this, the effects that each move has on AIF structures need to be carefully defined.

For a proponent's $Conjecture(c)$ and $Lemma(l_i)$ moves, all that needs to be done is to add c and the l_i to the AIF graph. The $ProofDone$ move is used to establish the

inferential connection between them (in AIF terms, it adds an RA-node with incoming edges from each of the l_i and an outgoing edge to the c). The inference is established by using proponent's commitment stores, connecting all those propositions in the *Lemmas* commitment store to that in the *Conjecture* commitment store.

An opponent's $Counter(m,c)$ move adds m to the graph, and introduces a conflict between m and c (in AIF this is captured as a CA-node connecting from m to c).

As an example of the proponent's possible responses to a *Counter*, *GlobalLemmaInc*orporation connects the opponent's counterexample, m with a specific lemma, introducing a new conflict, i.e. CA-node, between m and the specified l. This l is the lemma that must be replaced to fit in the hidden assumption and, as such, it is also removed from the *Lemmas* store. The proponent must next perform a *HiddenAssLemma* move to provide the previously missing hidden assumption. This new lemma, l_{hidden} (added to the AIF graph as an I-node) is added to the Lemma store and the proponent must next perform the standard Lemma move to replace the initially removed lemma. After which, the proponent can perform a *ProofDone* move with the current conjecture to add a new inference. The DGDL representing the specification of these moves can be seen in Listing 1.

Listing 1: Global Lemma Incorporation

```
interaction{GlobalLemmaInc, {l,m}, Contradicting,
    {<{m},{l}>, DefaultConflict}, "Remove lemma $l",
    {
        store(remove, {l}, Lemmas)
        & move(add, next, HiddenAssLemma,{h},proponent)
    }
}

interaction{HiddenAssLemma, {l}, Asserting, {l},
    "$l is a lemma",
    {
        store(add, {l}, Lemmas)
        & move(add, next, Lemma, {k}, proponent)
    }
}
```

AIF structures represent one way of handling 'structured argumentation.' Other approaches to structured argumentation, such as that provided by ASPIC+ [18], have been shown to be compatible, in that it is possible (under certain assumptions) to translate from AIF to ASPIC+. Prakken has further shown (*ibid.*) that ASPIC+ structures can be used to induce abstract argumentation frameworks which can make use of the wide range of existing argumentation semantics for computing acceptability.

The approach described in this section ensures that as a Lakatos Game is executed, AIF structures are created which, when translated to ASPIC+, produce abstract argumentation frameworks that under grounded semantics have as acceptable arguments all and only those elements which correspond to the proof accepted by the participants of the Lakatos Game.

Thus, determining the most up-to-date status of a proof in a Lakatos Game can be achieved using TOAST [13] for conversion to ASPIC+ and induction of abstract frameworks; and Dung-O-Matic [9] for calculation of acceptability semantics.

5. Further Work and Conclusions

While acknowledging Lakatos's contribution to the field, fellow philosophers of mathematical practice have criticised him for over-reliance on a few case studies, historical inaccuracy and narrowness of application of his theory [10].

Timothy Gowers recently initiated Polymath [2], a series of experiments in online collaborative mathematics, in which problems are posted online, and an open invitation issued for people to try to solve them collaboratively, documenting every step of the ensuing discussion. The resulting record provides an unusual example of fully documented mathematical activity leading to a proof. In [15], Pease and Martin present an analysis of one such project [1] and describe aspects of it which follow Lakatosian conversational patterns. They consider the following examples:

Example 1:

> Perhaps even the line does not matter! Is it possible to prove that any point and any line will do? (Anonymous, 8:31 pm)
> No, if you start with two points on the convex hull (ordered in the right way) you stay on the convex hull. (Thomas H, 8:35 pm)

Example 2:

> Ohhh... I misunderstood the problem. I saw it as a half-line extending out from the last point, in which case you would get stuck on the convex hull. But apparently it means a full line, so that the next point can be "behind" the previous point. Got it. (Jerzy, 8:31 pm)

In the first example the Anonymous contributor implicitly *surrenders* the conjecture that the line does not matter. In the second example, the relevant concept is a 'line', and a potential counterexample is raised. Participants argue that the counterexample is not valid, *monster-barring* it, and explain their reasoning, and the original participant responds by redefining sub-concepts in the conjecture, thus making the problem object a supporting example. These examples show that Lakatos-style reasoning can be used to describe real world examples of mathematical conversations.

We have formalised Lakatos's theory of mathematical discovery and justification as a dialogue game and then implemented our formalisation. Other than the system by Pease *et al.*, described in [16], this is the only implementation of Lakatos's dialogue moves that we are aware of. While there is a large body of work in Automated Theorem Proving, this is rarely based on ways in which humans do mathematics. Integrating automated theorem provers with Lakatos-style exploration could lead to a greater degree of flexibility (for instance, see [8]). Focusing on informal mathematics and on work which has a more "human-like flavour" is a challenging but valuable endeavour. We intend to continue this work along two main axes:

1. Capture more of Lakatos's theory, including concept-stretching and the method of proofs and refutations;

2. Run our LG on real world examples such as those described above and refine it accordingly.

If we are successful then this suggests a new area for dialogue games, capturing fundamental aspects of theories of mathematical and scientific discovery and justification. In addition to furnishing us with a greater understanding of how these processes operate, this would provide the basis for systems which can be used both to aid mathematicians in their work and to train students to think and interact in expert-like ways.

Acknowledgements

We are very grateful to our three anonymous reviewers for their useful and thought-provoking comments. We gratefully acknowledge the support of the Polish National Science Center under grant 2011/03/B/HS1/04559 and the support of EPSRC under grant EP/G060347/1. We also gratefully acknowledge the support of the COINVENT project, under FET-Open Grant number: 611553. This research was supported in part by the RCUK Lifelong Health and Wellbeing Programme grant number EP/K037293/1 - BE-SiDE: The Built Environment for Social Inclusion in the Digital Economy.

References

[1] Minipolymath3 project. http://polymathprojects.org/2011/07/19/minipolymath3-project-2011-imo/.
[2] The polymath blog. http://polymathprojects.org/.
[3] A. Aberdein. The uses of argument in mathematics. *Argumentation*, 19:287–301, 2005.
[4] A. Aberdein. The informal logic of mathematical proof. In B. v. Kerkhove and J. P. v. Bendegem, editors, *Perspectives on Mathematical Practices: Bringing Together Philosophy of Mathematics, Sociology of Mathematics, and Mathematics Education*, pages 135–151. Springer. Logic, Epistemology, and the Unity of Science, Vol. 5, 2007.
[5] J. Alcolea Banegas. L'argumentació en matemàtiques. In E. Casaban i Moya, editor, *XIIè Congrés Valencià de Filosofia*, pages 135–147. Valencià, 1998.
[6] F. Bex, J. Lawrence, M. Snaith, and C. Reed. Implementing the argument web. *Communications of the ACM*, 56(10):66–73, 2013.
[7] C. C. Chesñevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316, 2006.
[8] S. Colton and A. Pease. The TM system for repairing non-theorems. In *Selected papers from the IJCAR'04 disproving workshop, Electronic Notes in Theoretical Computer Science*, volume 125(3). Elsevier, 2005.
[9] J. Devereux and C. Reed. Strategic argumentation in rigorous persuasion dialogue. In *Proceedings of ArgMAS 2009*. Springer, 2009.
[10] P. Ernest. *Social constructivism as a philosophy of mathematics*. State University of New York Press, Albany, NY, 1997.
[11] E. C. W. Krabbe. Strategic maneuvering in mathematical proofs. *Argumentation*, 22(3):453–468, 2008.
[12] I. Lakatos. *Proofs and Refutations*. Cambridge University Press, Cambridge, 1976.
[13] S. M. and C. Reed. Toast: online aspic+ implementation. In *Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA 2012)*, 2012.
[14] A. Pease and A. Aberdein. Five theories of reasoning: Inter-connections and applications to mathematics. *Logic and Logical Philosophy*, 20(1-2):7–57, 2011.
[15] A. Pease and U. Martin. Seventy four minutes of mathematics: An analysis of the third mini-polymath project. In *Proc. of the AISB Symp. on Mathematical Practice and Cognition II*, pages 19–29, 2012.
[16] A. Pease, A. Smaill, S. Colton, and J. Lee. Bridging the gap between argumentation theory and the philosophy of mathematics. *Foundations of Science*, 14(1-2):111–135, 2009.
[17] H. Prakken. Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 21:163–188, 2006.
[18] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1, 2010.
[19] S. Toulmin. *The uses of argument*. CUP, Cambridge, 1958.
[20] S. Toulmin, R. Rieke, and A. Janik. *An Introduction to Reasoning*. Macmillan, London, 1979.
[21] S. Wells and C. Reed. A domain specific language for describing diverse systems of dialogue. *Journal of Applied Logic*, 10(4):309–329, 2012.