

*25th International Workshop on Computational Models of Natural  
Argument, Online, 12th December 2025*

---

# A Notation for Declaratively Describing Arguments

---

Simon Wells<sup>1</sup> & Mark Snaith<sup>2</sup>

<sup>1</sup> Edinburgh Napier University

<sup>2</sup> Robert Gordon University

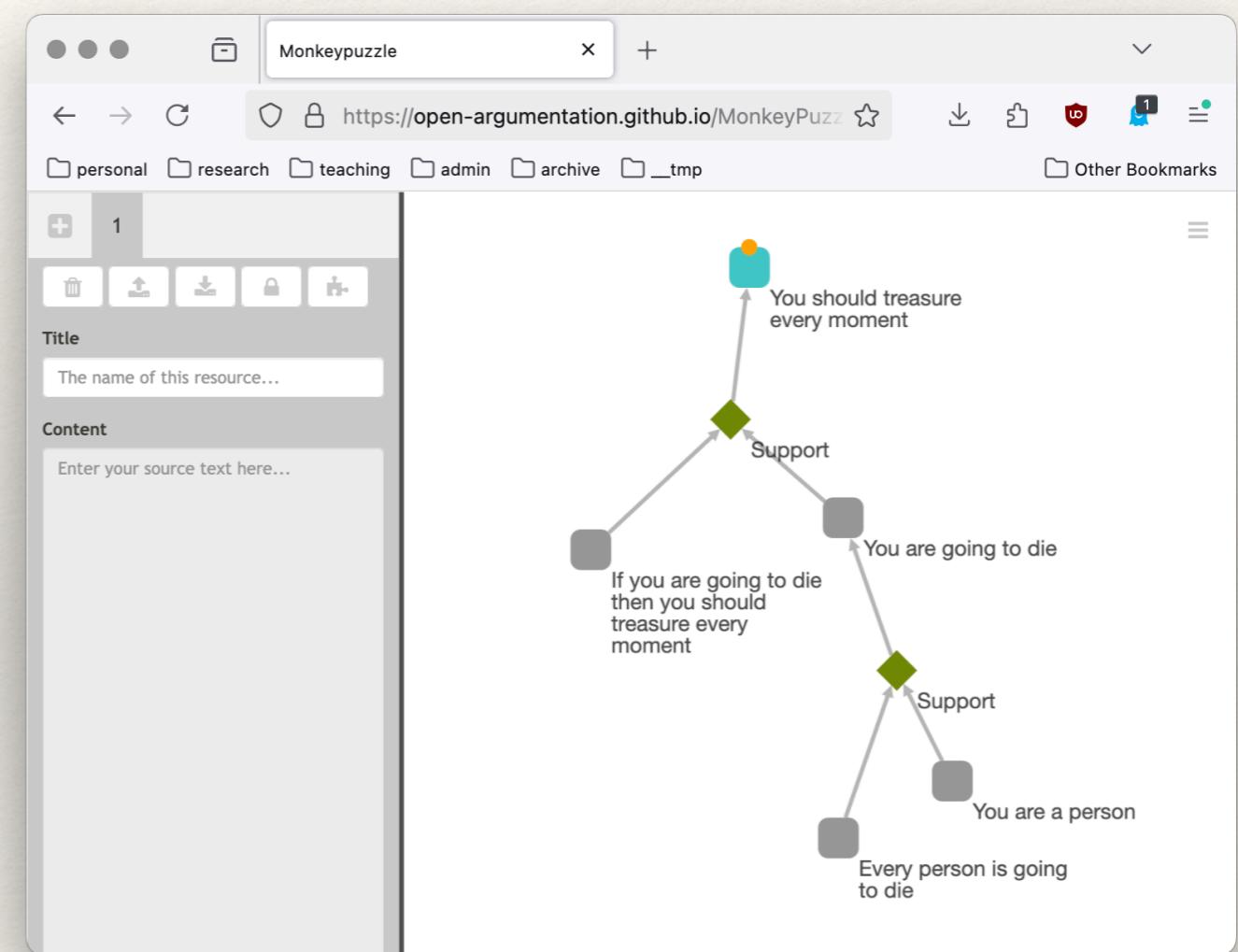
---

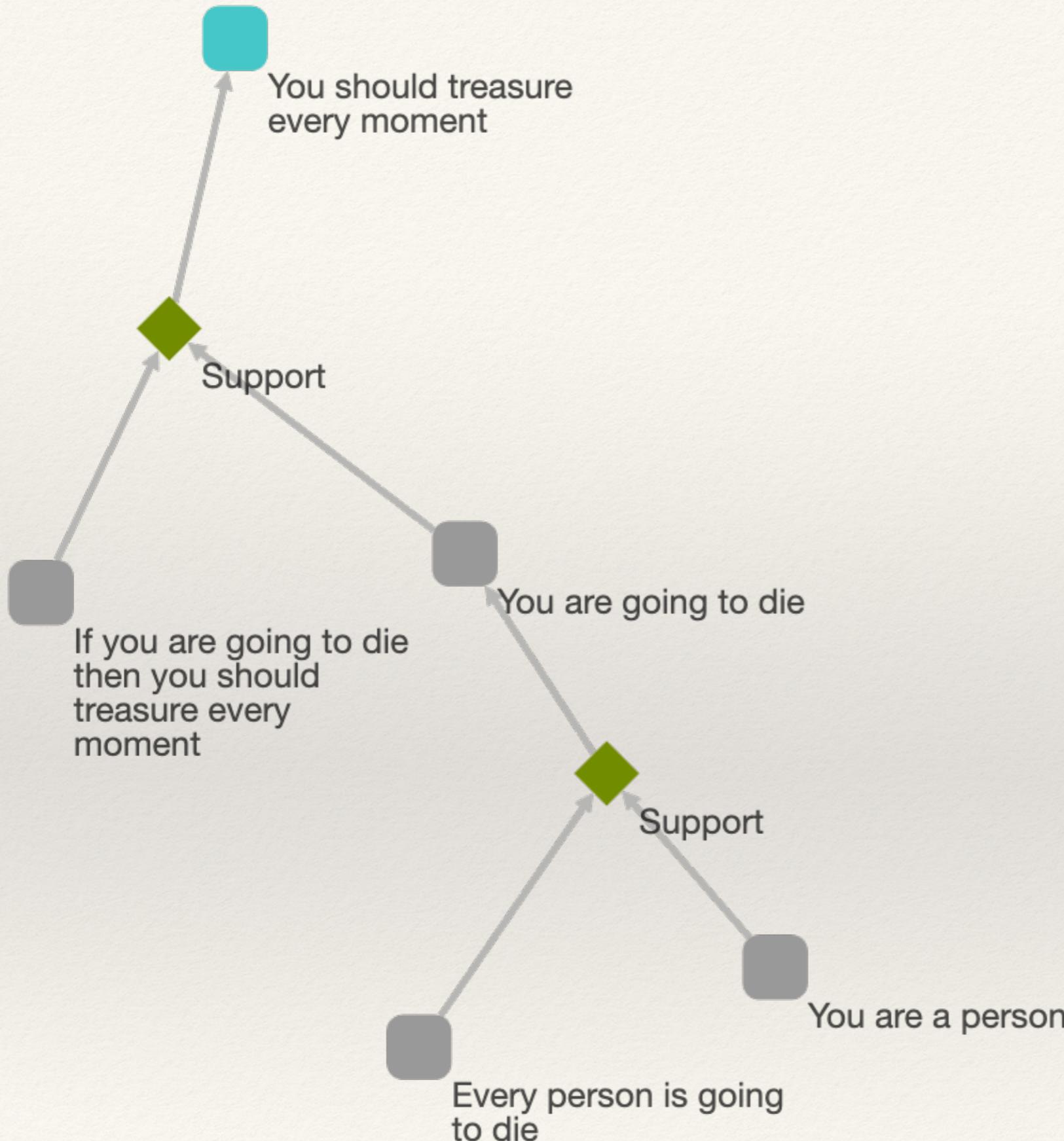
# Alternatively: A Tale of Two Technologies

- ❖ We've been working on a range of software tools to for working with arguments:
  - ❖ PreFACE
  - ❖ SADFace

# MonkeyPuzzle & SADFace

- ❖ MonkeyPuzzle was developed as a testbed for investigating ways to visually analyse arguments from multiple, multimedia, resources.
- ❖ Led to SADFace, the Simple Argument Description Format & ultimately to (OAPL) the Open Argumentation PLatform.

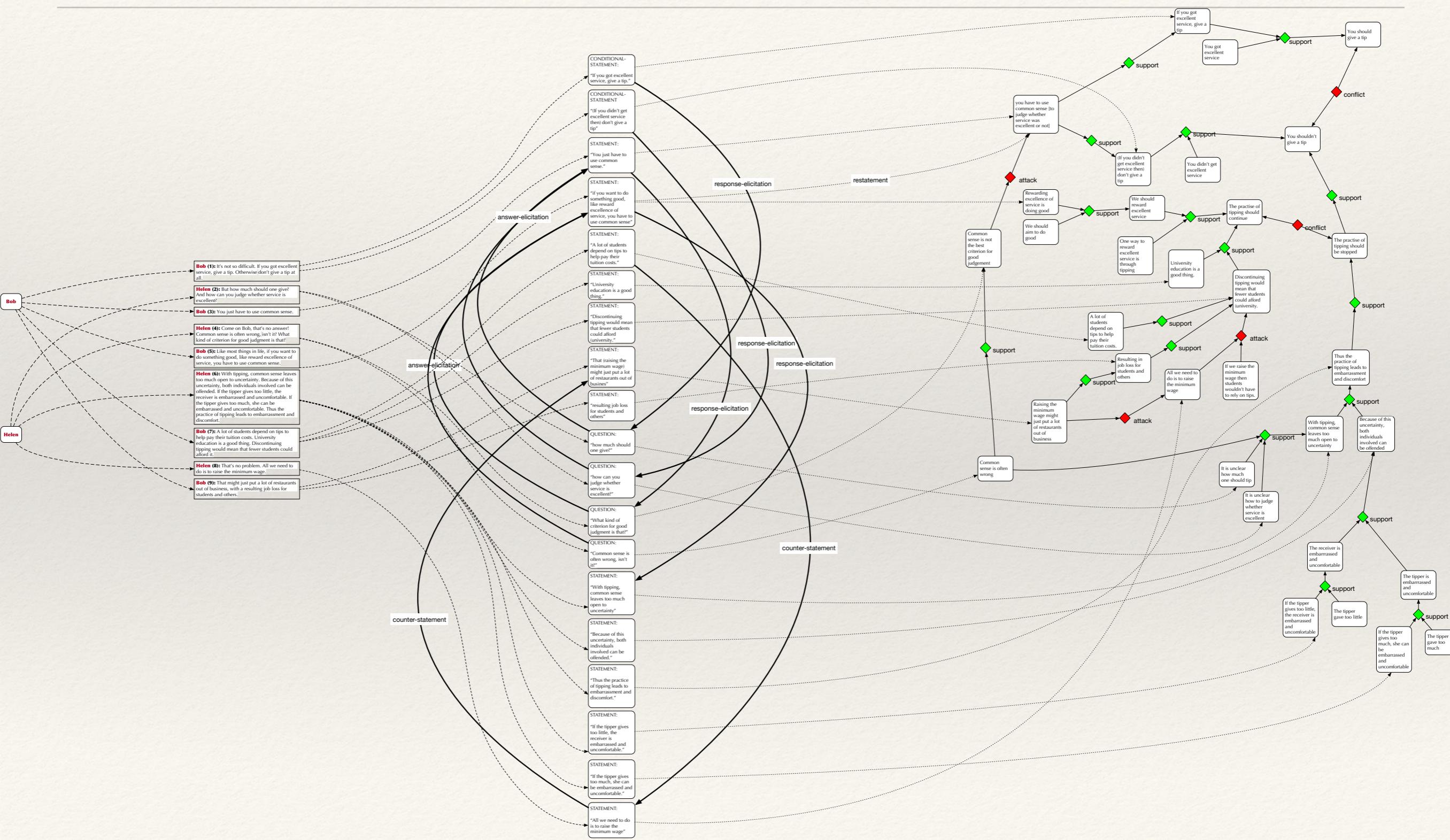




```

{
  "edges": [
    {
      "id": "a1s1",
      "source_id": "a1",
      "target_id": "s1"
    },
    {
      "id": "a2s1",
      "source_id": "a2",
      "target_id": "s1"
    },
    ...
  ],
  "metadata": {
    "core": {
      "analyst_email": "s.wells@napier.ac.uk",
      "analyst_name": "Simon Wells",
      "created": "2018-02-23T02:27:36",
      "edited": "2025-09-23T11:12:39",
      "id": "94a975db-25ae-4d25-93cc-1c07c932e2f9"
    }
  },
  "nodes": [
    {
      "id": "a1",
      "metadata": {},
      "sources": [],
      "text": "Every person is going to die",
      "type": "atom"
    },
    {
      "id": "a2",
      "metadata": {},
      "sources": [],
      "text": "You are a person",
      "type": "atom"
    },
    {
      "id": "s1",
      "name": "Support",
      "type": "scheme"
    },
    ...
  ],
  "resources": [
    {
      "id": "c4f04a87-79f6-479d-890d-496a89c9416f",
      "content": " ",
      "type": "text",
      "metadata": {}
    }
  ]
}
  
```

# Zonal ARgumentative DialOgue visualiZation (ZARDOZ)



# PreFACE

- ❖ A tool for regulating dialogues (agent-agent, agent-human, human-human).
- ❖ A framework for bringing together Dialogue Games (rich formal structure) & LLMs (good, human-level, generated utterances).
- ❖ Aim to overcome a range of issues from each technology:
  - ❖ structured dialogue with natural expression.
  - ❖ adaptive goal-oriented & goal-preserving behaviour.
  - ❖ Evidence-based output with explainability.
  - ❖ Formal structure, naturally phrased.
  - ❖ Grounded, domain-specific, multi-source knowledge.
- ❖ Essentially: an interface / wrapper around an LLM: provide scaffolding & context, generate prompt, evaluate result.

# Conversation (paraphrased [quite heavily])

---

- ❖ **Mark:** “I’ve been doing this thing for PreFACE and I’ve invented this other thing.”
- ❖ **Simon:** “Cool. That sounds like this thing I also built for SADFace to help with another thing.”
- ❖ **Mark:** “These things are really quite similar.”
- ❖ **Simon:** “Yes. If we’ve independently invented these two quite similar things then maybe others will find it useful?”
- ❖ **Both:** “Yes. Let’s share our toys with the other children.”

# SADN

- ❖ The Simple Argument Description Notation.
- ❖ A simple, declarative language for describing arguments.
- ❖ Builds on SADFace
- ❖ Where SADFace is document-oriented, SADN is incremental and declarative

# SADN Commands

- ❖ Two basic structures: simple (bare command) & complex (taking argument) of which argument are either simple (individual datums) or complex (key:value pairs):
  - ❖ atom(id:\$ID, text:\$TEXT)
  - ❖ scheme(id:\$ID, name:\$TEXT)
  - ❖ support(scheme:\$ID, conclusion:\$ID, premise:\$ID, ...)
  - ❖ attack(source:\$ID, target:\$ID)
  - ❖ disagree(\$ID, \$ID, ...)
  - ❖ argument(scheme:\$TEXT, conclusion:\$ID | text:\$CONTENT, premise:\$ID | text:\$CONTENT, ...)
  - ❖ remove(id:\$ID, ...)
  - ❖ metadata(id:\$ID, namespace:\$TEXT, \$KEY:\$TEXT)
  - ❖ atoms, schemes, conflicts , edges
  - ❖ export(type:\$TYPE)
  - ❖ save(type:\$TYPE, name:\$PATHNAME)
  - ❖ load(name:\$PATHNAME)
  - ❖ clear, print, prettyprint, help

# Aliasing

- ❖ SADFace is constructed around globally unique identifiers (UUIDs) for the overall analysis, as well as for individual nodes, edges, & resources within it.
  - ❖ e.g. "id": "b00535db-25b8-45b4-99e9-2955fa0ba54c
- ❖ These are unwieldy for human UI usage.
- ❖ So we use aliasing to map SADFace UUIDs to local identifiers for the following types: atoms (a), schemes (s), edges (e), and conflicts (c).
- ❖ Local IDs use the identifier letter code for the type as well as monotonically increasing integer IDs for each UUID, so a list of atoms might be [a1, a2, a3, ...]
- ❖ Internally SADN maintains lookup tables to enable the SADFace UUID to be retrieved for any local ID, and vice versa.
- ❖ Local IDs are not maintained between SADN sessions and are generated as necessary at runtime then discarded once the session ends. Only SADFace UUIDs persist.

# Worked Example

- ❖ Recreate the MonkeyPuzzle example argument using SADN to interactively declare the elements of the argument and build the resulting model in an incremental fashion.



```

{
  "edges": [
    {
      "id": "81d94ca5-4cb1-4c4d-8486-708d45b45c4e",
      "source_id": "a03ea3f2-70f2-42ed-962c-f15367e79de7",
      "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"
    },
    {
      "id": "233 af9c1-ba7e-4967-9c32-3025ac65ee67",
      "source_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999",
      "target_id": "bba487d5-029a-4a1b-ae79-ad69c769382a"
    },
    {
      "id": "75c9ef27-7ba4-4cbf-8fde -e957698f472b",
      "source_id": "6e73e177-4e62-45e4-bd0d-62d72891c339",
      "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"
    },
    {
      "id": "e21ddbcc-ce50-4ce9-839e-5f8f5e2d5af4",
      "source_id": "df8e97d1-b2d6-4768-bd82-ea91a8820313",
      "target_id": "42ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb"
    },
    {
      "id": "adbc1971-5334-4daa-86bb-d7d706a3b8cc",
      "source_id": "42 ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb",
      "target_id": "3b666117-d50c-4de8-b2ed-29 f6ae98f10a"
    },
    {
      "id": "00695abb-e8aa-4f08-9af8-7fca55d53270",
      "source_id": "bba487d5-029a -4a1b-ae79-ad69c769382a",
      "target_id": "42ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb"
    }
  ],
  "nodes": [
    {
      "id": "a03ea3f2-70f2-42ed-962c-f15367e79de7",
      "type": "atom",
      "text": "Every person is going to die",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "6e73e177-4e62-45e4-bd0d-62d72891c339",
      "type": "atom",
      "text": "You are a person",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "bba487d5-029a -4a1b-ae79-ad69c769382a",
      "type": "atom",
      "text": "You are going to die",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999",
      "type": "scheme",
      "name": "Support"
    },
    {
      "id": "df8e97d1-b2d6-4768-bd82-ea91a8820313",
      "type": "atom",
      "text": "If you are going to die then you should treasure every moment",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "3b666117-d50c-4de8-b2ed-29f6ae98f10a",
      "type": "atom",
      "text": "You should treasure every moment",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "42ba2acc-d8d2-4d8f-ae93- c7f8c383b4bb",
      "type": "scheme",
      "name": "Support"
    }
  ],
  "metadata": {
    "core": {
      "analyst_email": "Default Analyst Email",
      "analyst_name": "Default Analyst Name",
      "created": "2025-10-29T17:53:05",
      "edited": "2025-10-29T18:50:13",
      "id": "4ab333ea-a003-4631-afa7-5f20806a8497",
      "version": "0.5.3.2"
    }
  },
  "resources": []
}
  
```

- ❖ Start the SADFace REPL
- ❖ This gives access to an interactive environment where we can type SADN commands at the “>” prompt

```
The SADFace REPL. Type 'help' or 'help <command>' for assistance  
>
```

- ❖ We can use the prettyprint command (or ‘pp’ shorthand) to see the state of the SADFace model.
- ❖ Default values are generated and inserted if a configuration file is not supplied.

```
The SADFace REPL. Type 'help' or 'help <command>' for assistance
> pp
{
  "edges": [],
  "metadata": {
    "core": {
      "analyst_email": "Default Analyst Email",
      "analyst_name": "Default Analyst Name",
      "created": "2025-10-29T17:53:05",
      "edited": "2025-10-29T17:53:30",
      "id": "4ab333ea-a003-4631-afa7-5f20806a8497",
      "version": "0.5.3.2"
    }
  },
  "nodes": [],
  "resources": []
}
>
```

- ❖ We use the atom() command to introduce three new atoms to the model.
- ❖ We then use the print command to show the SADFace model
- ❖ Finally we use the atoms command to list the atoms introduced so far.

```

> atom(text:"Every person is going to die")
> atom(text:"You are a person")
> atom(text:"You are going to die")
> print
{
  "edges": [],
  "metadata": {
    "core": {
      "analyst_email": "Default Analyst Email",
      "analyst_name": "Default Analysyt Name",
      "created": "2025-10-29T17:53:05",
      "edited": "2025-10-29T17:53:30"
    },
    "id": "4ab333ea-a003-4631-afa7-5f20806a8497",
    "version": "0.5.3.2"
  }
}, {
  "nodes": [
    {
      "id": "a03ea3f2-70f2-42ed-962c-f15367e79de7",
      "type": "atom",
      "text": "Every person is going to die",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "6e73e177-4e62-45e4-bd0d-62d72891c339",
      "type": "atom",
      "text": "You are a person",
      "sources": [],
      "metadata": {}
    },
    {
      "id": "bba487d5-029a-4a1b-ae79-ad69c769382a",
      "type": "atom",
      "text": "You are going to die",
      "sources": [],
      "metadata": {}
    }
  ],
  "resources": []
}
> atoms
[{"a1": {"id": "a03ea3f2-70f2-42ed-962c-f15367e79de7", "text": "Every person is going to die"}, {"a2": {"id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "text": "You are a person"}, {"a3": {"id": "bba487d5-029a-4a1b-ae79-ad69c769382a", "text": "You are going to die"}]}

```

- ❖ We add a support relationship between the three atoms introduced thus far.
- ❖ This introduces an undifferentiated default “support” scheme between them.
- ❖ We use the schemes command to list the schemes introduced so far to the SADFace model.
- ❖ We then print the SADFace model to see the status of our model.

```
> support(conclusion: "a3", premise: "a1", premise: "a2")
>schemes
[{"s1": {"id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999", "name": "Support"}]
> print
{"edges": [{"id": "81d94ca5-4cb1-4c4d-8486-708d45b45c4e", "source_id": "a03ea3f2-70f2-42ed-962c-f15367e79de7", "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"}, {"id": "233af9c1-ba7e-4967-9c32-3025ac65ee67", "source_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999", "target_id": "bba487d5-029a-4a1b-ae79-ad69c769382a"}, {"id": "75c9ef27-7ba4-4cbf-8fde-e957698f472b", "source_id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"}], "metadata": {"core": {"analyst_email": "Default Analyst Email", "analyst_name": "Default Analyst Name", "created": "2025-10-29T17:53:05", "edited": "2025-10-29T18:02:25", "id": "4ab333ea-a003-4631-afa7-5f20806a8497", "version": "0.5.3.2"}}, "nodes": [{"id": "a03ea3f2-70f2-42ed-962c-f15367e79de7", "type": "atom", "text": "Every person is going to die", "sources": [], "metadata": {}}, {"id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "type": "atom", "text": "You are a person", "sources": [], "metadata": {}}, {"id": "bba487d5-029a-4a1b-ae79-ad69c769382a", "type": "atom", "text": "You are going to die", "sources": [], "metadata": {}}, {"id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999", "type": "scheme", "name": "Support"}], "resources": []}
```

- ❖ Finally we use the `argument()` command to introduce an entire support structure in a single command.
- ❖ This is a mix of existing local IDs and new text content.
- ❖ Atoms are created in the background to account for new texts.
- ❖ We then list atoms and schemes as before then print the model.

```

> argument(conclusion: "You should treasure every moment", premise: "a3", premise: "If
  you are going to die then you should treasure every moment")
> atoms
[{"a1": {"id": "a03ea3f2-70f2-42ed-962c-f15367e79de7", "text": "Every person is going to die
  "}, {"a2": {"id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "text": "You are a person"}, {"a3": {"id": "bba487d5-029a-4a1b-ae79-ad69c769382a", "text": "You are going to die"}, {"a4": {"id": "3b666117-d50c-4de8-b2ed-29f6ae98f10a", "text": "You should treasure every
  moment"}, {"a5": {"id": "df8e97d1-b2d6-4768-bd82-ea91a8820313", "text": "If you are
  going to die then you should treasure every moment"}}
> schemes
[{"s1": {"id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999", "name": "Support"}, {"s2": {"id": "42
  ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb", "name": "Support"}}
> print
{"edges": [{"id": "81d94ca5-4cb1-4c4d-8486-708d45b45c4e", "source_id": "a03ea3f2-70f2-42ed
  -962c-f15367e79de7", "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"}, {"id": "233
  af9c1-ba7e-4967-9c32-3025ac65ee67", "source_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999
  ", "target_id": "bba487d5-029a-4a1b-ae79-ad69c769382a"}, {"id": "75c9ef27-7ba4-4cbf-8fde
  -e957698f472b", "source_id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "target_id": "b3b0e13a-8b8b-4e2a-9544-5ab401e58999"}, {"id": "e21ddbcc-ce50-4ce9-839e-5f8f5e2d5af4
  ", "source_id": "df8e97d1-b2d6-4768-bd82-ea91a8820313", "target_id": "42ba2acc-d8d2-4d8f
  -ae93-c7f8c383b4bb"}, {"id": "adbc1971-5334-4daa-86bb-d7d706a3b8cc", "source_id": "42
  ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb", "target_id": "3b666117-d50c-4de8-b2ed-29
  f6ae98f10a"}, {"id": "00695abb-e8aa-4f08-9af8-7fca55d53270", "source_id": "bba487d5-029a
  -4a1b-ae79-ad69c769382a", "target_id": "42ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb"}], "metadata": {"core": {"analyst_email": "Default Analyst Email", "analyst_name": "Default
  Analyst Name", "created": "2025-10-29T17:53:05", "edited": "2025-10-29T18:50:13", "id
  ": "4ab333ea-a003-4631-afa7-5f20806a8497", "version": "0.5.3.2"}}, "nodes": [{"id": "a03ea3f2-70f2-42ed-962c-f15367e79de7", "type": "atom", "text": "Every person is going to
  die", "sources": [], "metadata": {}}, {"id": "6e73e177-4e62-45e4-bd0d-62d72891c339", "type
  ": "atom", "text": "You are a person", "sources": [], "metadata": {}}, {"id": "bba487d5-029a
  -4a1b-ae79-ad69c769382a", "type": "atom", "text": "You are going to die", "sources": [],
  "metadata": {}}, {"id": "3b666117-d50c-4de8-b2ed-29f6ae98f10a", "type": "atom", "text": "You should
  treasure every moment", "sources": [], "metadata": {}}, {"id": "df8e97d1-b2d6-4768-bd82-ea91a8820313", "type": "atom", "text": "If you are
  going to die then you should treasure every moment", "sources": [], "metadata": {}}, {"id": "42ba2acc-d8d2-4d8f-ae93-c7f8c383b4bb", "type": "scheme", "name": "Support"}], "resources": []}

```

# Applications

- ❖ Rapid, interactive, construction of arguments.
- ❖ TextUI for working with arguments.
- ❖ Interlingua for describing structured argumentative content, i.e. for use in LLMs and other ML.
- ❖ Investigating the narrative & temporal aspects of argumentative reasoning.
- ❖ Piping argumentative / debate data between tools.

# Future Work

- ❖ Implementation is nearly complete:
  - ❖ Mostly testing & fine-tuning the language.
  - ❖ SADFace is both mature & gaining traction (according to download stats) so intend SADN to be ready for active use upon release.
- ❖ Tree-based edit model (temporal development of arguments)
- ❖ Use of SADN within PreFACE.
- ❖ Describing dialogue (wider SADFace development goal).

# Conclusions

- ❖ Introduced SADN:
  - ❖ A notation for declaratively describing arguments for use in constructing, analysing, understanding, and mapping argumentative dialogue.

# Acknowledgements

- ❖ The authors wish to thank our reviewers for the kind, interested, and useful suggestions which will inform the future directions of this work.
- ❖ Partly supported by the Engineering and Physical Sciences Research Council (EPSRC) of the UK Government (grant number UKRI220).

# References

- ❖ M. Snaith & S. Wells (2021) "Towards a Declarative Approach to Constructing Dialogue Games" in proceedings of the 21st International Workshop on Computational Models of Natural Argument (CMNA'21)
- ❖ M. Snaith & S. Wells (2024) "Formal Dialogue and Large Language Models", presented at the SICSA Workshop on Reasoning, Explanations and Applications of Large Language Models (REALLM), Aberdeen, UK.
- ❖ M. Snaith & S. Wells (2025) "Retrieval Augmented Generation for Immersive Formal Dialogue" Post-proceedings of the 5th European Conference on Argumentation (ECA 2025), *forthcoming*
- ❖ S. Wells and Douglas J. Monkeypuzzle: Towards next generation, free & open-source, argument analysis tools. In Proceedings of the 17th International Workshop on Computational Models of Natural Argument (CMNA17), 2017.
- ❖ S. Wells (2020) "The Open Argumentation PLatform (OAPL)" in Proceedings of Computational Models of Argument (COMMA 2020), Frontiers in Artificial Intelligence, Volume 326: Computational Models of Argument, pages 475-476, Perugia, Italy.