"Lifted Planning for Semi-Abstract Interactive Storytelling" Spring 2018 Final Project Progress document.

Magnús Ágúst Magnússon
magnusm15@ru.is

Arnar Páll Jóhannsson
arnarpj15@ru.is

Sævar Óli Valdimarsson
saevarv15@ru.is

May, 2018

Contents

Preface

The following paper is a compilation of several related papers, all concerning the Spring 2018 final project *"Lifted Planning for Semi-Abstract Interactive Storytelling"*, a project being worked on in conjunction with CADIA by the authors of this paper. The reports are mainly concerned with the ongoing process and progress of the project and acts as a detailed log, laying the foundation of the management of the project. The documents are a combinations of descriptions, plans, logs and accounts of the team as they work on the project, and they aim to give a high level overview of the progression of the project over a given time period.

Part I
Lifted Planning for Semi-Abstract IS: Description

1 Project summary

1.1 Abstract

The SAGA project is an ongoing research project in the context of interactive storytelling (IS), where a simulated environment is dynamically created, monitored, and modified to produce meaningful narrative experiences for one or more players. An interactive storytelling system is a reactive system that can control and adapt to unexpected situations created by players within the environment, toward maintaining the stories that occur in the virtual world. The SAGA system is an interactive storytelling system that uses an AI planner to determine story events. For this work, we propose to improve upon this planner allowing it to accept more general inputs and by understanding (and perhaps improving) its current abstraction model in terms of lifted representation planning.

1.2 Detailed description

The project aims to be completed in two parts:

The first part of the proposed project seeks to expand the capabilities of SAGA's planner, allowing it to accept generalized predicates and non-binary relations. The current system only allows unary or binary relations, which limits both author creativity and the potential complexity of outlines. By expanding SAGA's capacity to handle relations, we hope to make the SAGA system better able to handle complicated storylines and make it more comfortable for authors to use. The project aims to refactor the codebase of the Saga system where needed and redesign the internal classes, databases and generators so that they are better fit to accept an unspecified amount of entities in terms of its relations and predicates.

The second part of the proposed project seeks to explore the concept of Lifted Representation Planning (LRP), toward better understanding its similarities and differences from SAGA's current model of abstraction, and how its underlying theory can be applied in the context of the SAGA project. Based on this investigation, additional features based on LRP will be devised and implemented as part of the SAGA system. Finally, a research paper will be prepared that summarizes the findings of this work.

1.3 Progress plan

The project is split into 5 phases.

Phase 1:
- Prepare working conditions
- Study the SAGA system
- Study GOLOG
- Read related research material

150 hours

Phase 2: Implement generalization to non-binary predicates. 150 hours.

Phase 3: Research Lifted Representation Planning. 300 hours.

Phase 4: Implement LRP-based improvements. 150 hours.

Phase 5: Research Paper on planner improvements. 150 hours.

1.4 Deliverables and project result

1.4.1 Learning Outcomes

It is expected of the team members that over the course of the project they shall attain a much deeper understanding of planning AI. They should also get increased knowledge and proficiency of GOLOG and increased skills in writing professional research reports.

1.4.2 Deliverables

The team aims to deliver a more effective and general version of SAGA's planning system. The new system should allow for more complicated relations, ability to accept more general predicates, and at least one or more features based on lifted representation planning.
Along this the team shall deliver a research report detailing their research, work and discoveries in direct relation to the SAGA system, lifted representation planning, and improvements made to the system.
The team should also turn in a report detailing their overall final project progress, work, and other logged data unrelated to SAGA but related directly to their work-structure and logged information collected over the course of the 15 week project.

1.4.3 Evaluation

The new planner will be evaluated on the basis of its completeness in terms of meeting the project's objectives, as well as how well its new code can be understood, both from the code itself and from any accompanying documentation. The research paper will be evaluated on the basis of how completely it describes the work and on the quality of its preparation.

2 Work Structure

2.1 Work environment

The group will have its regular meetings and working area in the CADIA offices located on the second floor of Reykjavík University.

2.2 Work Hours

The group has decide that each team member should, in some way or form, work on the project for an average amount of 20 hours per week. it is the responsibility of the team member in question to make sure that adequate work is being put into he project on his behalf.
The group has scheduled regular meetings three times per week: Wednesdays, Thursdays and Fridays; from 12:30 to 18:00. Length of meetings may change based on workload.

Other days are free and are used based on practicality and prior commitments, but team-members are encouraged to allocate a time on their own to work on the project.

2.3 Scrum

The group has decided upon the Scrum framework to plan our work-flow and organize tasks. The project owner is David Thue and Scrum master is Arnar Páll Jóhannsson as he is the team-member with the most Scrum experience. We have decided upon 2 week sprints that start and end on Fridays. The length of the sprints may change during the final exam period as well during the 3 weeks after the finals. Scrum meetings will be held on the later parts on Fridays.

2.4 Equipment

No equipment or physical hardware is required for the development of the project aside from the computers supplied by the students for their own use.

Non-physical equipment and services to be used by the students are the following.

1. Trello
2. Toggl
3. Sharelatex
4. Google Docs/Sheets/Slides
5. Git
6. Java
7. Golog
8. Eclipse
9. Unity
10. SWI-Prolog

3 On changed situations and conflict

In the case of conflict between this report and other reports the report whose main topic is the issue of conflict holds priority. If both reports have equal claim to the issue the newer of the two reports holds priority, as it should reflect the more recent design ideas of the group. The group holds the authority to alter report priority if they find so to be required by a 2/3 majority of the group.

In the case of conflict between team members the team should try to resolve the issue by themselves. In the failure of civil discussion a simple majority of the team members shall decide the issue. In case of this procedure being ineffective or for large-scale conflicts the supervisor or project owner shall have final say.

4 General backlog and risks

4.1 Goal backlog

Below is a brief overview over the tasks the project will require us to complete. The table is roughly ordered in the order of expected completion. The "Estimated sprint" column is a rough estimate of the sprint we'll be working on said goal or completing it in case of long term goals. The status then indicates the current status of that goal, be it completed, ongoing or not started.
The index is color coded. A green cell indicates that this goal has been completed and will not need revisiting under normal circumstances.. Red cells indicate the goal has been cancelled and won't be completed this project.

ID	Goal	Estimated sprint	Current status
1	Unity setup	0	Completed in S0
2	Eclipse setup	0	Completed in S0
3	Project proposal	0	Completed in S0
4	Peer presentation	1	Completed in S1
5	Reading material	1	Completed in S1, S8
6	Status meeting 1	2	Completed in S2
7	Learn the structure of the SAGA system	2	Completed in S2-S6
8	Implement 3 way predicate	2-3	Cancelled.
9	Test 3 way predicate	3	Cancelled
10	Refactor required classes for generalization	3-4	Completed in S2
11	Refactor relevant databases	3-4	Completed in S3
12	Refactor any API calls to generalized classes	3-4	Completed in S3
13	Generalize GOLOG predicates	4	Completed in S4
14	Status meeting 2	4	Completed in S4
15	Test generalized predicates	4-5	Completed in S5
16	Debug and cleanup implemented code	5	Completed in S6
17	Study lifted representation planning	5-7	Completed in S7
18	Report on LRP and see what can be used in SAGA	6-7	Completed in S9
19	Implement LRP in SAGA	5-7	Cancelled
20	Finish semester report	7-9	Completed in S9
21	Write final report	7-9	Completed
22	Status meeting 3	9	Completed
23	Final presentation	7-9	Completed

4.2 Risk assessment

Every risk below is composed of two factors: Probability and an Impact factor.
The first is an indicator from 1 to 3 that measures how likely we believe that this particular risk is to happen. A measure of 1 means we think that the risk will never happen, while a risk of 3 means we are certain that the risk will happen at least once over the course of the project.
The work impact factor is an indication with the scale of 1 to 3 on the amount of disruption the particular risk is to our project in the case that it happens in terms of workload, i.e how much work is required for us to fix the result of the risk and maintain the overall project. A measure of 1 means that we are fairly sure that the event will have minimal, if any, impact on our work. A risk of 3 indicates that this risk will prevent us from completing that project in a satisfactory manner without inhumane effort.

The time impact factor is an indication with the same scale on the amount of disruption the particular risk is to our project in the case that it happens in terms of time, i.e how long would it take us to fix the cause of the risk. A measure of 1 means that we could fix it in negligible time. A risk of 3 indicates that this risk will most likely not be fixable given the time we're given.
The Risk factor is then a scale from 1 to 9 indicating the total danger of that risk, and is calculated as the three other factors multiplied together. The higher the risk factor is, the more danger we believe comes of this risk and thus the more we need to prevent it from happening, as well as place more effort into a contingency plan to minimize damage if this risk occurs. A factor of 1 means we think this risk will not have any impact on the project and does not need to be defended against. A factor of 9 means that the risk is highly probable and will have high impact on the project. Those risk are high priority problems that need to be prevented or at least mitigated. The scale follows then a roughly linear scale from these two extremes, albeit the values should be judged with some contextual interpretation, seeing as all measures are rough judgment calls made by us based on rough gut feeling.

- Any risk on the scale 1-4 should be considered as a mild threat.
- Any risk equalling 6 should be considered a significant threat and must have a proper prevention measures and contingency plans.
- Any risk equalling 9 is a very serious danger and must receive highest priority and care. The risk must be especially considered in all actions made in the projects, and prepared against as well as a detailed plan formed to minimize the damage.

ID	Risk	Probability (1-3)	Impact (1-3)	Risk factor (1-9)
3	We are unable to implement generalized pred. on prolog side	2	3	6
5	There is no place where LRP can be implmented	2	3	6
7	We start the final report(s) too late	2	3	6
8	We manage to break the Saga system	2	3	6
10	The report source is lost	2	3	6
13	We cannot understand the SAGA system	2	3	6
1	The code behaves in an incorrect way	2	2	4
4	We are unable to understand LRP	2	2	4
6	Implementing LRP in code is too complex	2	2	4
11	We get too distracted with a low-priority task	2	2	4
14	We cannot write a high-quality report	2	2	4
15	Other classes take up too much work/attention	2	2	4
16	Setup becomes difficult or for some reason fails	2	2	4
17	Our solution is suboptimal	2	2	4
9	A team member drops out	1	3	3
19	Someone gets sick	3	1	3
20	The entire codebase gets destroyed in a fire	1	3	3
12	We are unable to work 900 man hours	2	1	2
18	We run out of things to do	1	2	2
2	We are unable to implement generalized pred. on java side	1	1	1

Part II
Final report

5 Overview

The following report is intended as an overview of the work performed by the group over the course of the project. The following sections are split into sprints: Two week work sessions focused on a specific goal. Each sprint section will specify what we hoped to accomplish during said sprint, a time table listing the raw hours worked, difficulties faced during the sprint as well as a discussion on the sprint where we will discuss what we accomplished and felt we could do better.
In an ideal sprint we hoped to collectively work 120 hours. We expected to accomplish the main goal of the sprint or finish it to a degree so that we can say that we have gained a significant understanding of the goal and have a good way of completing it later in the project. As much of the project revolved around research and gaining a thorough understanding of the research material, some goals didn't have a definite end-point but rather at a certain point we as a team agree that we understood the material sufficiently such that we could move on to move on to other parts of the project.

6 Sprint 0

The zeroth sprint lasted from the Friday of 12th of January to the Friday of 26th of January.

6.1 Goal

The goal of sprint 0 was general setup, getting introduced to the project, getting access to all repositories, writing the project proposals for the research council, and initializing all systems required for the project and its development. Preparation of documentation was also a main focus of sprint 0.

6.2 Time Table

	Arnar	Magnús	Sævar	Total
12th of Jan	0:00	0:49	0:00	0:49
13th of Jan	0:00	0:00	0:00	0:00
14th of Jan	0:00	0:00	0:00	0:00
15th of Jan	0:00	0:00	0:00	0:00
16th of Jan	0:00	0:00	0:00	0:00
17th of Jan	3:33	3:32	3:33	10:38
18th of Jan	2:52	0:00	0:00	2:52
19th of Jan	1:29	3:45	3:30	8:44
20th of Jan	0:00	0:00	0:00	0:00
21st of Jan	0:00	0:00	0:00	0:00
22nd of Jan	0:00	0:00	0:45	0:45
23rd of Jan	0:00	0:14	0:00	0:14
24th of Jan	3:33	4:01	3:20	10:54
25th of Jan	4:11	2:50	4:41	11:42
26th of Jan	0:00	1:58	0:00	1:58
Total	15:40	17:12	15:50	48:43

6.3 Disucssion

The 0th sprint was fairly uneventful, albeit successful. As the goal was to set up the project and the required systems, as well as getting ourselves acquainted with the task at hand.
The better part of the sprint was spent in writing proposals and project reports. Through writing the proposal we got a glimpse of what we would be doing and what was expected of us. We were granted access to the SAGA system and given a short guide on how to set it up. While most of us could get eclipse running and the SAGA system TCP server running comfortable without much trouble, the Unity setup provided some greater challenge. We agreed that the inflated size and pointlessly detailed world scene did more to harm than help.
It took a bit of time to get everything to a point such that we could start development, but overall we believe that the goals of the sprint were sufficiently completed.

7 Sprint 1

The first sprint lasted from the Saturday of 27th of January to the Friday of 9th of February

7.1 Goal

The goal of sprint 1 was reading the source material, preparing for the first lecture, and in general get familiar with the core theory behind the Saga system.

7.2 Time Table

	Arnar	Magnús	Sævar	Total
27th of Jan	0:00	0:00	0:00	0:00
28th of Jan	0:00	0:00	0:00	0:00
29th of Jan	0:00	0:00	0:00	0:00
30th of Jan	0:00	0:00	0:00	0:00
31st of Jan	4:28	4:16	4:14	12:58
1st of Feb	3:14	3:45	4:20	11:19
2nd of Feb	0:00	3:30	0:00	3:30
3rd of Feb	0:00	0:00	0:00	0:00
4th of Feb	0:00	0:00	0:00	0:00
5th of Feb	0:00	0:00	0:00	0:00
6th of Feb	2:00	2:00	2:00	6:00
7th of Feb	4:06	4:14	4:06	12:26
8th of Feb	3:50	3:15	3:20	10:25
9th of Feb	0:00	0:00	0:00	0:00
Total	17:38	21:00	18:00	56:38

7.3 Discussion

The core of this sprint was reading and performing independent research therefore it was not easy to properly quantify the level of success obtained within the sprint. There were no real hurdles to speak of this sprint. However during this sprint we did get a greater understanding of the theories supporting the SAGA system. We've had a minor discussion on the various reports we've printed out as part of the reading list. We've gone over the subjects of narrative trajectories, POV of charaters and motives in an IS environment, dynamic event ordering, and the transition of the system from PDDL to GOLOG.

8 Sprint 2

The second sprint lasted from the Saturday, 10th of February to the Friday 23rd of February

8.1 Goal

The main goal of sprint 2 was laying down the groundwork of predicate generalization. We aimed to refactor the Relation.java class to allow it to store a list of entities instead of being limited to three. We also refactored all classes that inherit from Relation so they would know how to operate their new found generalized constructors. We also took a look at the GOLOG side of things and removed a lot of redundant code. We also prepared for Status meeting 1, in the second half of the sprint.
The general goal was therefore to reduce clutter and lay down groundwork for implementing the first generalized predicates, the task of which is the main goal for the next sprint, sprint 3.

8.2 Time Table

	Arnar	Magnús	Sævar	Total
10th of Feb	00:00	00:00	00:00	00:00
11th of Feb	00:00	00:00	00:00	00:00
12th of Feb	00:00	00:00	00:00	00:00
13th of Feb	00:00	00:00	00:00	00:00
14th of Feb	02:05	03:09	01:59	07:13
15th of Feb	04:49	05:29	04:41	14:59
16th of Feb	03:34	00:00	03:39	07:13
17th of Feb	00:00	00:00	00:00	00:00
18th of Feb	00:00	00:00	00:00	00:00
19th of Feb	00:00	00:00	00:00	00:00
20th of Feb	00:00	00:00	00:00	00:00
21th of Feb	04:59	03:25	04:33	12:57
22th of Feb	05:28	03:21	04:35	13:24
23th of Feb	01:59	00:47	02:12	04:58
Total	22:54	16:11	21:39	60:44

8.3 Discussion

The general purpose of this sprint was refactoring the relations class, reducing GOLOG clutter and preparing for status meeting 1. All went according to plan without many issues. We found that the GOLOG basic script was riddled with redundancies or other oddities that we felt were of little use. We argued that the current state of the GOLOG code was the result of transposing it 1:1 from PDDL as a lot of the patterns we observed didn't fully utilize GOLOG. All in all we cut out a fair portion of repeated code out of that script.
The Relation class was successfully refactored. The class formerly stored three entities *Entity1, Entity2, Entity3*. We implemented a list of entities to replace the three entities and their types, along with providing a generalized API for the use of other classes. The several functions and constructors designed to interact with the old entities within the Relations class were made to interact via the generalized constructors, but otherwise were left in peace for a short while until we schedule a full-scale API refactor. This ensures that the rest of the project can go on via the old API without changing behavior. We even managed to get a head start on the next sprint by starting the refactor of the databases.
A good amount of whatever time remained was spent writing the documents and preparing a presentation required for the first status meeting, which had been significantly delayed. We had been stitching the report together bit by bit over the first two sprints and therefore we didn't have to touch it up alot to get it meeting-ready.
Not a lot of obstacles or barriers were met in this sprint. We netted a fairly average number of hours for this sprint, which isn't exactly what we were hoping for but we are working to mend this in future sprints.

9 Sprint 3

The third sprint lasted from the Saturday, 24th of February to the Friday 9th of March

9.1 Goal

The main goal of sprint 3 was the overall refactoring of the saga project with respects to the relation class, the legacy API that it used, the classes that derive from it, and the classes that interact with it.

In sprint 2 the Relations class was refactored, giving it the capacity to operate and store an undetermined number of entities. However, neither the constructors of the relations class family, nor the API, was changed to preserve functionality of other classes. This sprint revolved around tossing those out, and replacing every instance and appearance of them throughout the entire SAGA project with a more appropriate, generalized, version that strengthened the project as a whole.
There were ca. 400 constructor calls relying on the old design, so we estimated that at least a single whole work session was spent on nothing but grunt work replacing one constructor pattern for another.
We also kept on refactoring and generalizing the databases that represent relations in some way or form, and finally took a stab at the Prolog communicator and generator in an attempt to simplify its n^3 loops and make them general without $O(n!)$ complexity.

9.2 Time Table

date	Arnar	Magnús	Sævar	Total
24th of Feb	00:00	00:00	00:00	00:00
25th of Feb	00:00	00:00	00:00	00:00
26th of Feb	00:00	00:00	00:00	00:00
27th of Feb	00:00	01:53	00:00	01:53
28th of Feb	05:11	05:11	05:05	15:27
1st of March	04:05	05:26	03:48	13:19
2nd of March	03:28	05:33	01:30	10:31
3rd of March	00:00	00:00	00:00	00:00
4th of March	00:00	00:00	00:00	00:00
5th of March	00:00	00:00	00:00	00:00
6th of March	00:00	00:00	00:00	00:00
7th of March	05:08	04:40	05:01	14:49
8th of March	04:48	06:38	05:10	16:36
9th of March	04:37	04:50	03:55	13:22
Total	27:17	34:11	24:29	85:57

9.3 Discussion

The sprint brought on good news and bad news. The good news are that most of the goals we set out to complete this sprint managed to get completed, within acceptable reason. A full workday was spent going over every single constructor call referencing a relation and changing it to use the new, general, form of the constructor. The databases were finalized and the Prolog communicator was mostly generalized with minimal noticeable change in output.
After the changes we seem to have introduced a logic error somewhere. The system, while not throwing an error initially, would not find a plan for a given test scenario. This is a deviation from the behaviour of the system when we got control of it. It is therefore obvious that somewhere along the way our generalization changed the behaviour of the system, instead of preserving it as we had aimed for. It is of importance that this error is found and corrected before further work is put in place.

10 Sprint 4

The fourth sprint lasted from the Saturday, 10th of March to the Friday 23rd of March

10.1 Goal

The main goal of sprint 4 was a general dissection of the Golog planner. We had discovered a bug in our work that caused either no plan to be given or variations of the exact same plan be given multiple times. Among tasks was clearing out some obsolete or dead Prolog code from the planner; merging common, repeated patterns into a single general pattern; completing the generalization process of various functions that generate Prolog in the Java side of SAGA, and prepare for status meeting 2.

10.2 Time Table

date	Arnar	Magnús	Sævar	Total
10th of March	00:00	00:00	00:00	00:00
11th of March	00:00	00:00	00:00	00:00
12th of March	00:00	00:00	00:00	00:00
13th of March	00:00	00:00	00:00	00:00
14th of March	04:36	04:41	04:38	13:55
15th of March	05:07	04:52	04:48	14:47
16th of March	04:04	04:50	04:44	13:38
17th of March	00:00	00:00	00:00	00:00
18th of March	00:00	00:00	00:00	00:00
19th of March	00:00	00:00	00:00	00:00
20th of March	00:00	00:00	00:00	00:00
21st of March	03:06	04:29	04:52	12:27
22nd of March	03:28	03:14	04:16	10:58
23rd of March	00:00	03:03	03:38	06:41
Total	20:21	25:09	26:56	72:26

10.3 Discussion

We determined the source of the bug, it was a fault in the heuristic used by the planner. Prolog allows for non-deterministic reasoning which caused the heuristic give multiple values to a potential plan, meaning a plan was accidentally evaluated as multiple plans. We managed to mitigate that somewhat but completely fixing the bug would require rewriting the heuristic and potentially a significant portion of the planner itself. We deemed that outside of the project's scope, as it would have take time away from the core focus of the project. The sprint was otherwise a success. We finished clearing out the obsolete code and prepare for the status meeting.

11 Sprint 5

The fifth sprint lasted from the Saturday, 24h of March to the Friday 13th of April

11.1 Goal

The main goal of sprint 5 was to wrap up the generalization we made to the planner and test them. We have made many changes to the planner over the semester such as removing old unused code, refactoring and adding support for generalized predicates. This needed to be examined to ensure that the parts we touched upon would still work.

11.2 Time Table

date	Arnar	Magnús	Sævar	Total
24th of March	00:00	00:00	00:00	00:00
25th of March	00:00	00:00	00:00	00:00
26th of March	04:05	00:00	00:00	00:00
27th of March	03:28	00:00	00:00	00:00
28th of March	03:00	03:42	00:00	06:42
29th of March	02:24	01:09	00:00	01:09
30th of March	00:00	00:00	00:00	00:00
31st of March	02:35	01:13	00:00	03:49
1st of April	00:00	00:00	00:00	00:00
2nd of April	00:00	00:00	00:00	00:00
3rd of April	00:00	00:00	00:00	00:00
4th of April	05:06	04:26	05:01	14:35
5th of April	02:05	03:46	03:16	09:08
6th of April	04:41	04:44	03:53	13:19
7th of April	00:00	00:00	00:00	00:00
8th of April	00:00	00:00	00:00	00:00
9th of April	03:30	00:00	04:02	07:32
10th of April	00:00	00:00	00:00	00:00
11th of April	04:50	04:44	00:00	09:34
12th of April	04:06	04:42	00:00	08:49
13th of April	04:07	00:00	03:59	08:06
Total	36:28	28:30	20:13	85:12

11.3 Discussion

The generalized predicates were tested and found out to work. We could create a simple story that included 3-way and 4-way predicates that the planner could use to find a cohesive narrative. There are though problems with the generalized predicates. As they describe relations between more entities, more permutations are possible which results in a bigger state space. They give more freedom to the author to construct the stories as he wants but as a result more responsibility is placed on the author to ensure that the planning doesn't become to complex due to huge state space. This can be mitigated by a good heuristic and planning algorithm.

While examining the planner and testing it, we noticed a feature that had been overlooked in the generalization. This was because the example story we were working with did not utilize that feature. After discovering the feature we generalized it immediately.

12 Sprint 6

The sixth sprint lasted from the Saturday, 14h of April to the Saturday 21st of April

12.1 Goal

The main goal of sprint 6 was researching lifted representation planning and finalizing reports and documents. We had papers and articles that mentioned LRP from our instructor and had found online.

12.2 Time Table

date	Arnar	Magnús	Sævar	Total
14th of April	00:00	00:00	00:00	00:00
15th of April	00:00	00:00	00:00	00:00
16th of April	05:35	06:27	05:33	17:36
17th of April	05:20	06:04	06:40	18:05
18th of April	04:57	05:58	06:24	17:20
19th of April	00:00	00:00	05:00	05:00
20th of April	00:00	03:00	05:34	08:34
21th of April	02:47	00:00	02:13	05:00
Total	18:40	21:30	31:27	71:38

12.3 Discussion

We read articles and papers that found regarding Lifted Representation Planning (LRP). In many of the papers LRP was only mention by use and no greater explanation gone into it, neither theory or implementation. In other papers it was explained but still not in great detail. But we managed to get an general idea how LRP was defined and how to implement it. This sprint marks our switch to Writing-mode as all our implementations should be finished by this point.

13 Sprint 7

The seventh sprint lasts the week of 22nd of April to the 28th of April

13.1 Goal

Sprint 7 focuses on two main tasks:

Further gaining a deeper understanding of LRP and how it relates to the SAGA system. here we are mostly trying to find interesting facets of LRP that we believe would be a good fit in SAGA and we have a somewhat clear idea of the benefit and potential theoretical implementation of the given feature. Another facet we're looking into is how to reframe the already existing SAGA system and expressing it in terms of lifted representation. There are several ideas from LRP that we believe are already present in the system, and thus said features might become interesting hooks into further integrating LRP philosophy into SAGA.

The second task we are aiming for in this sprint is to wrap up our work on generalized predicates. This involves documentation of code we have not already documented, and write a short summary at the end of the progress report. This section should act as a 'final report' of sorts for the generalized predicates and be a brief overview of the changes we made and potential ideas for future work.

13.2 Time Table

date	Arnar	Magnús	Sævar	Total
22nd of April	00:00	00:00	00:00	00:00
23rd of April	04:59	06:03	07:34	18:37
24th of April	05:55	06:09	07:05	19:10
25th of April	06:14	03:55	07:12	17:23
26th of April	05:37	04:44	06:18	16:39
27th of April	04:35	06:13	06:03	16:51
28th of April	04:06	02:36	03:17	09:59
Total	31:28	29:42	37:31	98:42

13.3 Discussion

The 7th sprint was a great success. it represented the most man hours put into any single sprint of the project to date. Most of the week was spent in discussion, reading research papers on topics that might or might not be useful to our report, and going citation hunting in search of a better definition of LRP. We started plotting down ideas of implementations in for system that we might want to pursuit. Our research paper finally started to take some form, although it was directionless and more a collection of unrelated topics than an actual paper at this current point. The main goal was to get the structure down then refine them later. Lack of order wasn't a particular concern to us as long as we got the thoughts down on paper in some way or form. The latter half of the week allowed us to start forming the coherent narrative for the paper.
In this sprint we also focused on more or less putting a nice bow on generalized predicates and we feel that we've closed the topic fairly well outside of the final presentation. We documented the code to the best of our ability and allowance of time, and wrote the "Shortly on generalized predicates" chapter of this report, detailing our work in very short words and some potential ideas of future work.

14 Sprint 8

The eigth sprint lasts a week spanning from the 29th of April to the 5th of May.

14.1 Goal

Sprint 8 is the second to last sprint before handing in the final project, and the last sprint before the final project.
The goal of the sprint was twofold: Define a clear cut direction and topic to our paper. We will want to get done with as much of the paper as possible, if not finish it to a degree where all that is left is rewording, polishing, and improving. Alongside that we will have to start and more or less finish the presentation for Status meeting 1, and alongside that get a final presentation out of the deal, since we've established that both the final presentation and the status meeting presentation will be more or less identical save a few process-related details we won't require from the final presentation.

14.2 Time Table

date	Arnar	Magnús	Sævar	Total
29th of April	00:00	00:00	00:00	00:00
30th of April	05:58	05:52	05:54	17:25
1st of May	04:19	06:00	03:18	13:39
2nd of May	05:42	04:32	06:33	16:48
3rd of May	07:01	05:35	07:43	20:20
4th of May	06:12	06:22	07:45	20:21
5th of May	02:06	03:21	05:23	10:51
Total	31:03	31:45	36:38	99:27

14.3 Discussion

Sprint 8 went very well with very minimal obstacles being met. After a whole lot of reading and further citation hunting we have a very clear idea of what we think Lifted Representation Planning involves and how we might use it in the system along with examples relating to Robin Hood.
We've met with David both last sprint and this one, and his guidance along with the feedback he gave us in the report we sent to him allowed us to get a better idea of what we wanted to discuss in the paper. It might still be a little bit disjoint and we're still moving a bit from one topic to the other with unneeded abruptness, but we're moving into the right direction and have a more or less "ok" paper that, while not perfect, is a great groundwork for the final sprint.
We sent the paper to David for a second evaluation on Thursday and used the last two days of the sprint (Friday, Saturday) to work on the presentation and prepare for status meeting 3. On Saturday evening the slides were more o r less complete, and we had a very short live demo ready. What remains now of the presentation is deciding on the actual words we'll be saying on top of those slides and practice. The Sunday of the 6th will be spent on that task.

15 Sprint 9

The ninth sprint lasts a week spanning from the 6th of May to the 10th of May.

15.1 Goal

Sprint 9 is the last sprint before handing in the final project.
The goal of this sprint was to prepare ourselves for the status meeting on Wednesday, rewrite the progress report so it read more as a final report and then finalize the paper.

15.2 Time Table

date	Arnar	Magnús	Sævar	Total
6th of April	04:18	04:54	03:52	13:06
7th of April	08:27	07:31	09:38	25:38
8th of May	05:43	05:46	06:23	17:53
9th of May	06:16	12:16	10:02	28:35
10th of May	14:30	12:45	12:44	40:00
Total	39:17	43:14	42:41	125:13

15.3 Discussion

Sprint 9 marked an exciting last week of report writing. The week started off with a dry run of our presentation for David, where he provided us with various helpful pointers and critique. We spent the first couple of days preparing for the final status meeting, deciding properly what we wish to convey in the presentation and practice.
Once the status meeting was over in which we did not receiving any serious warnings about its content, we put our full focus on completing the final paper. We had received a reviewed draft from David which we then poked as many holes in ourselves as we could muster. From there we more or less restructured and rewrote the entire paper in just over three days.

16 Overall progress

16.1 Time Table

Sprint	Arnar	Magnús	Sævar	Total
Sprint 0	15:40	17:12	15:50	48:43
Sprint 1	17:38	21:00	18:00	56:38
Sprint 2	22:54	16:11	21:39	60:44
Sprint 3	27:17	34:11	24:29	85:57
Sprint 4	20:21	25:09	26:56	72:26
Sprint 5	36:28	28:30	20:13	85:12
Sprint 6	18:40	21:30	31:27	71:38
Sprint 7	31:28	29:42	37:31	98:42
Sprint 8	31:03	31:45	36:38	99:27
Sprint 9	39:17	43:14	42:41	125:13
Total	258:55	268:39	273:34	801:09

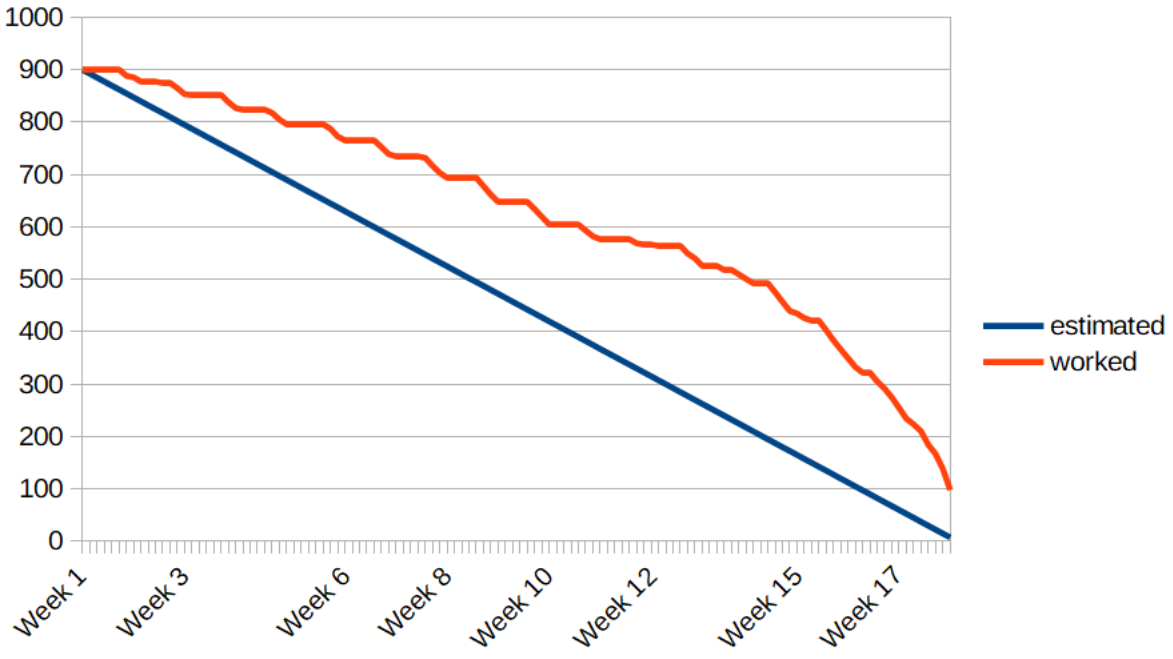
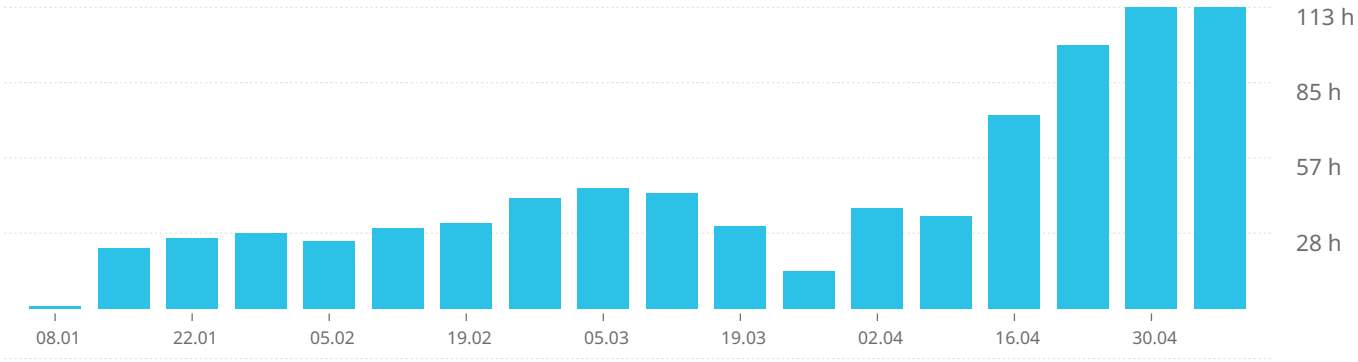


Figure 1: Total hours worked vs originally estimated hours worked

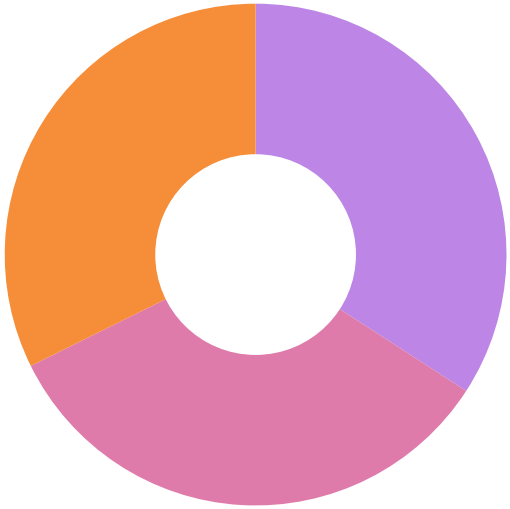
Summary report

2018-01-09 - 2018-05-11

Total 801 h 09 min

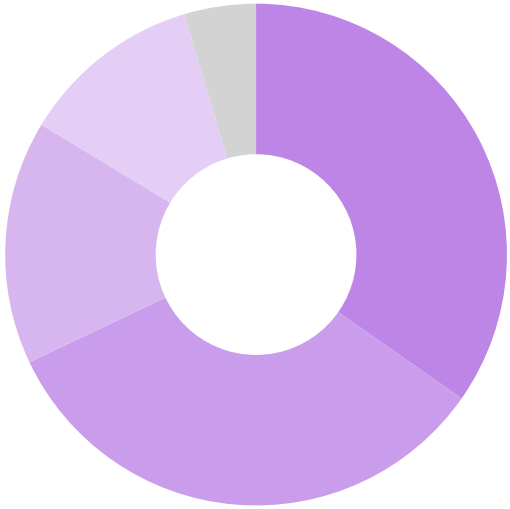


Users



- Sabbisun 273:34:24
- Magnusm15 268:39:42
- Arnarp96 258:55:26

Projects



- Reports 278:08:55
- Development 266:11:10
- Presentations 125:52:55
- Research 93:56:00
- Other 37:00:32

Part III
Shortly on generalized predicates

Since the research paper is set to feature lifted representation planning we felt it would be out of place to try to cover the generalization of the system alongside it. While the final code is fairly well documented it was decided to include a small section here as a overview of this part of the project.

17 Overview

Our generalization can be split into four sections: Relations, Databases, GOLOG Generation, and the planner.

17.1 Relations

In the former version of SAGA, relations were represented by a java class that contained a fixed number of entities, more specifically three. The entire system always acted as every relation had exactly three entities, and then just compensated by checking if any of the entities where, what we called, the "Empty entity", simply represented by the empty string. This was a weird anti-pattern, which both restricted the systems ability to generalize while at the same time providing no obvious benefit other than the knowledge that you'd always have exactly three entities, but at the cost of having to actually test if a given entity is empty when appropriate, instead of just having a variable number of entities and building the core logic around that.
One of the first things we did was to replace the three hard coded entities in the relation class with a list and a general API. Now, instead of relations assuming to be of a specific length the system can simply test the number of entities present in the relation and take appropriate action based on that.
All constructors of classes derived from Relation (or Relation itself) no longer can just specify 1, 2 or 3 entities and the constructors collapse and overload on themselves to fill in the blanks, but must present a list of entities (or more commonly two lists, one with entity names, one with entity types) in the constructor. In general we used the method Arrays.asList(entity1,entity2,entity3...).
The Relations base class now provides an API that allows for fetching specific entries from its list (be it a name, or a type), fetching the entire list as is, or getting the entire list of entities as ENTITY instances, hopefully allowing for smoother use.
All comparing functions have been update to reflect our changes, but received minimal testing due to time constraints.

17.2 Databases

The databases had been designed under the same preconceptions as the java system, relations and data-types that derived from relations could only contain 3 entities. This meant all tables contained columns for 3 entities, which if a relation only specified 1 or 2 entities, the rest would be filled with "Empty entities".
To generalize the predicates we went through the databases and applied the following methods.

- Identify a table or storage scheme where entities are already being stored globally for that database, if such a table does not exist create it.
- Wherever a relation mentions some entity extract the reference into a many to many linking table, connecting the relation in question with the entity table while also storing the position of the entity in the relation. This ensured that no information about the relations was lost.

That resulted in generalized databases that made it possible to have a relation entry connect to any number of entities.

17.3 GOLOG generation

A majority of our work went into the GOLOG generation, more specifically the GologCommunicationPlanner.java file. The core of the change was identifying every instance where relations are used in some hard coded way and generalizing them. For most functions where relations appeared the code was split into two parts: A unary case, and a binary case, which for the most part served the same function but (as their names imply), one only allowed a single entity to partake in the generated GOLOG while the binary case allowed two of them.
While each block of code and every function was slightly different the general process was as follows: start by identifying the function of the code, no point in trying to mend something you don't know the purpose of. After a general idea of the purpose and structure was obtained the conversion to a general case was started. Starting with merging the binary and the unary case and then further allowing "additional" entities mostly by removing hard coded statements and replacing them with dynamic loops.
Several sections had to be completely reimagined more or less from scratch. A clear example of such an section was the function "createActionMapCauses", which writes out the intended effects / causes of a given action.
In the original version the relevant pseudocode would look something like this:

```
Unary:
    For all entities x in action
        For all effects E mentioning x
            Generate GOLOG code for E(x).

Binary:
    For all entities x in action
        For all entities y in action
            if x != y
                For all effects E mentioning x,y
                    Generate GOLOG code for E(x,y).
```

This is easy to remember and works well enough, but has an obvious expansion problem. For every entity we wish to add to this chain we would have to add another loop. This both causes an $O(n^m)$ time complexity, given n possible entities for a m-way relation, but it also prevents us from properly implementing true generalization as we'd have to precede on the maximum number of loops we're willing to write.
The solution came by reversing the thought process: instead of checking all combinations of entities and see if there is some effect that matches, instead we should check all effects and see if we have the required entities for it.

```
N-ary:
    For all effects E of action
        For an ordered list L of entities mentioned in E and are present in action
            Generate GOLOG code for E(L)
```

This causes some initial overhead and pre-processing so we can properly extract and map the entities of the action to the entities of the effect, as we now have to "manually" arrange and identify the entities, their internal positions in the relation, and the existence of any "anonymous" entities not mentioned in the action. However, we now have the benefit that effects that need N-ary relations do not need a function that exponentially scales with N, but instead is strictly dependant on the number of entities present in the action.
Overall each function in the GOLOG planner communicator had a similar structure and issue: A loop or segment of code assuming a hard count of the entities expected to interact with that loop or segment, and most

of which required some restructuring and introduction of general loops to add in entities without knowing beforehand how many would or wouldn't appear.

17.4 Planner

When we got the planner into our hands it had a lot of redundant or unused code which we either rewrote or removed all together. The main suspected cause of a majority of this dead, redundant code was to be found in Tryggvi Þór Guðmundssons paper on Flexible Authoring using GOLOG Planning (2017). In his research it appears that he had mostly performed a 1:1 translation between PDDL and GOLOG, resulting in code that was functional and correct, but did not fully utilize the strengths and properties of GOLOG. This lack of proper porting left us with correct, but bloated code that we could remove and rewrite as mentioned.
We also noticed that at some points in the execution of the planner it left non-deterministic choice points where it shouldn't do so. In this process we ran into a problem where the planner generated the same plan over and over again instead of giving us new and unique solution. Sadly we couldn't correct this completely because of time restraints so the planner still might generate the same plan multiple times.

18 Potential future work

18.1 On the importance of constraints

A core problem with planning systems is that without proper care the search space quickly grows to intolerable sizes, dragging the entire system down to a halt. The importance of constraining what characters can partake in what actions or what entities can be imbued with what qualities cannot be overstated. Assume for instance N 2-way actions with M entities in the world. none of the actions have any constraints or conditions on what entities can partake in them. That would leave us with $N \times M^2$ potential steps that could be taken at a given moment in the story. For smaller stories this is perfectly doable, but it does not take a long time before it starts overburdening the system for long or complicated story spaces. A 7 step plan with just 3 possible actions and 4 entities still results in a search tree with 5.87×10^{11} potential states.
However, this problem is drastically worse now that authors have the ability to define general predicates and actions. Given N Z-way actions over M entities the planner can now take $N \times M^Z$ potential steps during planning. The planner now has to deal with exponentially more potential outcomes and actions every step of the planning process.
While we developed our extensions of the Saga system we became aware of the importance of strict conditions and limitations for the actions given. If an action or predicate can only choose three specific entities as targets instead of the entire entity space of M entities, or specify that only the 'hero' defined in the outline can participate, or that the abstract entity must be bound to a concrete entity before becoming a valid target, then it will not become overburdened as quickly, and the system might have a fighting chance of generating a good story before running out of memory. However the more you limit the potential steps the planner can take the more restricted authoring you put on the author, which might make it a chore to write stories for the system.
It is our suggestion of a future project to look into researching and developing a better and clearer way of restraining actions, outlines and relations. We believe that authors should be given powerful, yet easy to use, tools that allow them to define their story in a precise way that allows the system to operate at reasonable speeds without forcing the author to limit his stories or take away from the joy of creation. The system should benefit both the author, as well as the planner, to ensure an optimal generated narrative in a reasonable time-frame.

18.2 A quick optimization of databases

Currently the databases store a list of entities in one table, a list of relations in another, and a third table linking the two together. However, the databases for the most part only do the bare minimum to prevent redundancy. The main culprit is the entities tables, which usually won't check if two entities are the same in all aspects outside the primary key. The system could be extended to identify this recurrence alongside other possible violations of normality. While the overall effect of these 'violations' are insignificant and are only a question of organized, clean data entry, it might be a good project for someone over a weekend or two.

18.3 Reworking the GOLOG planner

As we mentioned above in the planner section we had a problem where we generate the same plans multiple times unfortunately solving this problem was not within scope of our project as we didn't have time to spend on this. We managed though to determine a part of the reason for this error. It was how the way how a best first search algorithm in the GOLOG planner was implemented. GOLOG is an extension of Prolog, which mainly uses constraint satisfaction to solve problems. A GOLOG query can have multiple solutions, if it finds one solution it backtracks in the constraint satisfaction problem to find another solution. This is a useful functionality but it has to be used correctly. The best first search uses a heuristic which had non-deterministic parts so it gave multiple solutions. When the GOLOG planner found a solution and backtracked to find another it would backtrack in the heuristic instead of the best first search part. Cutting down the non-determinism in the heuristic and ensuring that the backtracking in the search function as designed should fix these problem. There might be other problems that also contribute to this and the GOLOG planner has to be scrutinized more.

18.4 Various other improvements

Patterns and reappearing concepts are a big part in literature and other storytelling medium. The evil stepmother and the rule of three are just a few examples of many prevalent in storytelling. These patterns and concepts are often recognized by the viewer/player/user and invoke some preconceptions. Being aware of them gives the author new tools to either use or subvert.
Another improvement to the system would be to allow action preconditions to include the state of abstract roles. This would be necessary if an author would want to only allow specific roles to perform certain actions or to guide concrete entities to behave in a way that reflects on their assigned role. An example of this would be that only the abstract role "chosen one" would be able to kill "monster". This would allow the author to create outlines with stronger effect on the narrative generation. Admittedly the current tools would be sufficient to ensure a narrative shown in the example through intermediary goal but allowing different level of constraints gives the author more fine-grained control.