My CAREER NSF Journey

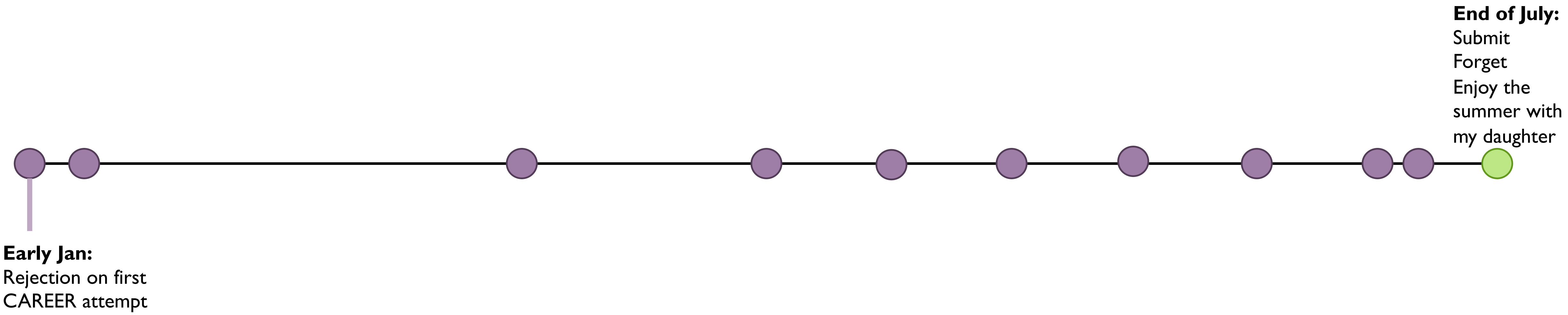
Two attempts and a baby later

Allison K. Sullivan, Ph.D.

My Career in a Nutshell – The NSF Perspective

- Assistant Professor at The University of Texas at Arlington – currently in 4th year
- **Research umbrella:** software engineering, formal methods
- **Division:** SHF
- **CAREER NSF Path**
  - **Attempt 1:** Summer between 2nd and 3rd year
  - **Attempt 2:** Summer between 3rd and 4th year (funded – starts June 2024)
- **Prior NSF Funding:**
  - CORE Small Grant – solo PI
  - FmitF Track II (x2) – solo PI
  - MRI – co-PI
- **Served on NSF Panels:** 5 before CAREER submission

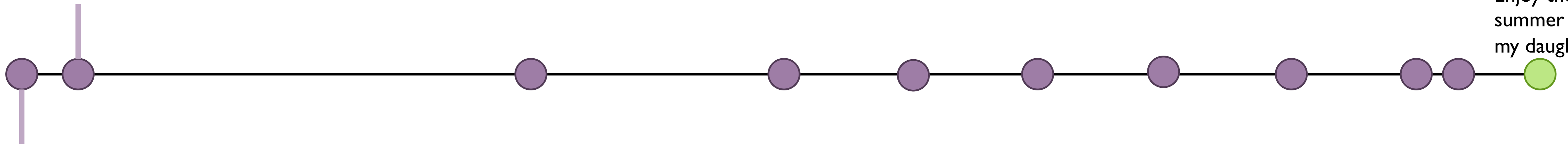
Timeline


End of July:
Submit
Forget
Enjoy the summer with my daughter
Early Jan:
Rejection on first CAREER attempt

Timeline

Early Jan:
My daughter is born!!
Also my first child

End of July:
Submit
Forget
Enjoy the summer with my daughter



Early Jan:
Rejection on first CAREER attempt

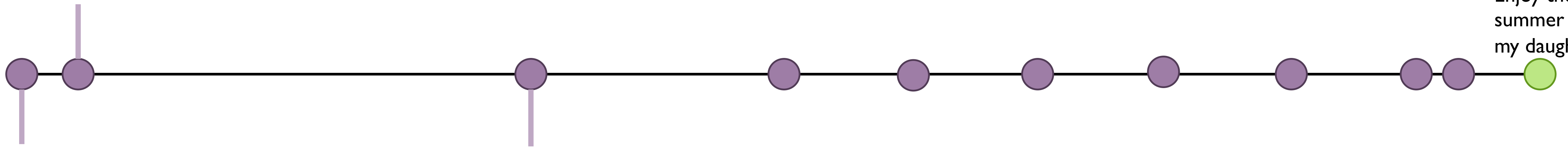
Timeline

Early Jan:
Rejection on first
CAREER attempt

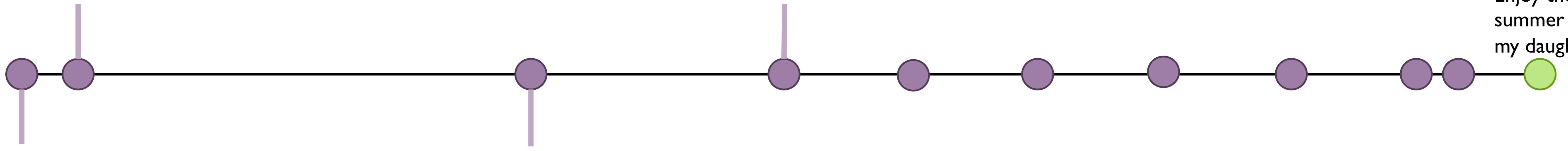
Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

End of July:
Submit
Forget
Enjoy the
summer with
my daughter



Timeline



Early Jan:
Rejection on first
CAREER attempt

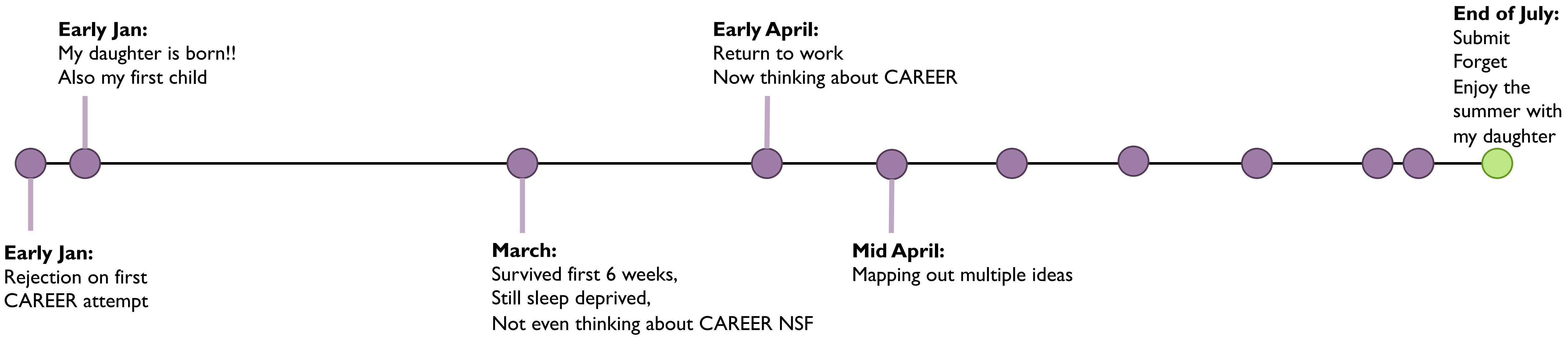
Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Timeline



Early Jan:
Rejection on first
CAREER attempt

Early Jan:
My daughter is born!!
Also my first child

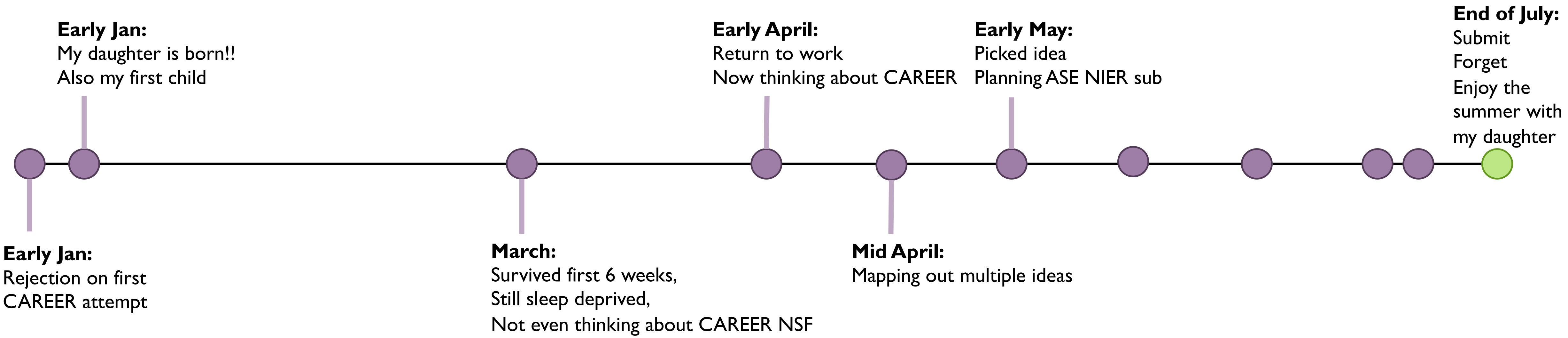
March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

Mid April:
Mapping out multiple ideas

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Timeline



Early Jan:
Rejection on first
CAREER attempt

Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

Mid April:
Mapping out multiple ideas

Early May:
Picked idea
Planning ASE NIER sub

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Timeline

Early Jan:
Rejection on first
CAREER attempt

Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

Mid April:
Mapping out multiple ideas

Early May:
Picked idea
Planning ASE NIER sub

Mid May:
ASE NIER drafted
Starting proposal draft

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Timeline

Early Jan:
Rejection on first
CAREER attempt

Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

Mid April:
Mapping out multiple ideas

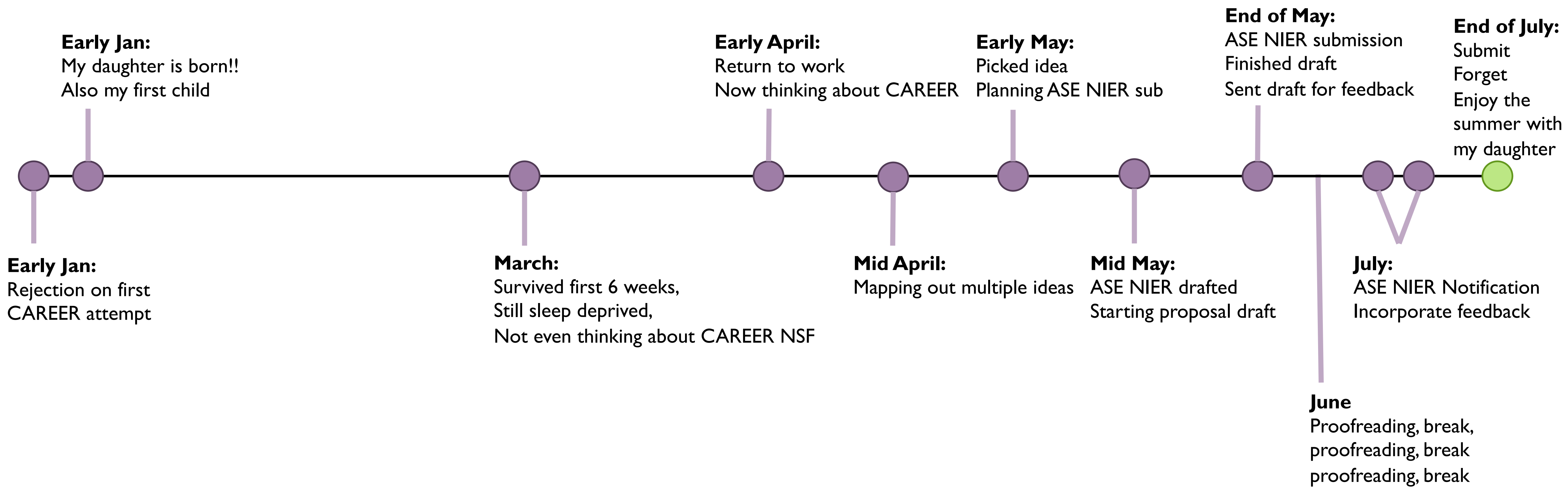
Early May:
Picked idea
Planning ASE NIER sub

Mid May:
ASE NIER drafted
Starting proposal draft

End of May:
ASE NIER submission
Finished draft
Sent draft for feedback

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Timeline



Early Jan:
Rejection on first
CAREER attempt

Early Jan:
My daughter is born!!
Also my first child

March:
Survived first 6 weeks,
Still sleep deprived,
Not even thinking about CAREER NSF

Early April:
Return to work
Now thinking about CAREER

Mid April:
Mapping out multiple ideas

Early May:
Picked idea
Planning ASE NIER sub

Mid May:
ASE NIER drafted
Starting proposal draft

End of May:
ASE NIER submission
Finished draft
Sent draft for feedback

June
Proofreading, break,
proofreading, break
proofreading, break

July:
ASE NIER Notification
Incorporate feedback

End of July:
Submit
Forget
Enjoy the
summer with
my daughter

Switching Topics

- **First attempt:** Low Competitive
  - First thoughts after submission: *is this just a CORE grant idea?* x_x
  - Reading between the lines of the reviews:
    - Excitement about thrust 2
    - Not too hot on thrust 1
    - Could I rework thrust 1? *Maybe but the core approach wouldn't really change*
    - Thrust 2 dependent on thrust 1

So, I decided to brainstorm down a new direction

Switching Topics

- **Brainstorming a new idea:**
  - **Thought process:**
    - What is the real goal I want for my research? (high level theme)
    - How could I try to tackle this goal leaving my old approach behind?

Switching Topics

- **Brainstorming a new idea:**
  - **Thought process:**
    - What is the real goal I want for my research? (high level theme)
    - How could I try to tackle this goal leaving my old approach behind?
  - **My result:**
    - **My goal:** get more people to use software models
    - **Old approach (*rooted in dissertation*):** software testing → automated repair → synthesis
    - **New approach:** live programming

I had no prior experience with live programming – but I was confident in my ability to explore it within the context of the modeling language I had been using (expertise concerns)

Switching Topics

- **Brainstorming a new idea:**
    - **Thought process:**
        - What is the real goal I want for my research? (high level theme)
        - How could I try to tackle this goal leaving my old approach behind?
    - **My result:**
        - **My goal:** get more people to use software models
        - **Old approach (*rooted in dissertation*):** software testing → automated repair → synthesis
        - **New approach:** live programming

I had no prior experience with live programming – but I was confident in my ability to explore it within the context of the modeling language I had been using (expertise concerns)

How I felt brainstorming this topic:
"How does this not exist yet?"
"This feels like an entirely new direction for my work but still feels like a natural progression"

[*I liked my first attempt and assumed I'd go back to that work but its not even on my radar now*]

Proposal Structure

My Official Outline

1. Introduction (2 pages)
    - 1.2 Research and Education Plan
    - 1.2 Broader Impacts

2 Background (1.75 pages)
- 2.1 Limitations of Writing a Model
- 2.2 Limitations of Exploring and Correcting a Model

3 Research Thrusts (9.25 pages)
- 3.1 Core Techniques
    - 3.1.1 Formula Completion Suggestions
    - 3.1.2 Enumeration View
    - 3.1.3 Focus View
- 3.2 Bidirectional Transformation Techniques
    - 3.2.1 Output Directed Debugging
    - 3.2.2 Solution-Driven Base Models
- 3.3 New Applications
    - 3.3.1 Lynx: Live Programming IDE for Pardinus
    - 3.3.2 Modeling-ABCs: Interactive Educational Tool for Mathematical Logic

4 Related Work (.75 pages)

5 Education Plan (1 page)

6 Project Plan (.25 pages)

7 Results from Prior NSF Support (.25 pages)

Scoping the Project

It is a 5 year grant but the budget supports one well funded PhD student

Main concept
- 3 key features – a papers worth of work each
- For me: 3 different live interfaces

Follow Up – Once main concept exists, what is enabled?
- Generally, 3 x 3 but not all research thrusts have 3 divisions

Toolsets embodying the research
- Education & Evaluation connection
- Outline notable support for usability/adoption/expansion

Proposal Structure

<u>My Official Outline</u>

1. Introduction (2 pages)
    - 1.2 Research and Education Plan
    - 1.2 Broader Impacts

2 Background (1.75 pages)
- 2.1 Limitations of Writing a Model
- 2.2 Limitations of Exploring and Correcting a Model

3 Research Thrusts (9.25 pages)
- 3.1 Core Techniques
    - 3.1.1 Formula Completion Suggestions
    - 3.1.2 Enumeration View
    - 3.1.3 Focus View
- 3.2 Bidirectional Transformation Techniques
    - 3.2.1 Output Directed Debugging
    - 3.2.2 Solution-Driven Base Models
- 3.3 New Applications
    - 3.3.1 Lynx: Live Programming IDE for Pardinus
    - 3.3.2 Modeling-ABCs: Interactive Educational Tool for Mathematical Logic

4 Related Work (.75 pages)

5 Education Plan (1 page)

6 Project Plan (.25 pages)

7 Results from Prior NSF Support (.25 pages)

<u>My Unofficial Outline</u>

Preliminary Work
Formula Template Suggestions.
Suggestion Boxes.

<u>My Unofficial Outline</u>

Limitation of Prior and Related Work.
Solution Editor.
Fault Localization.
Edit Suggestions.

Proposal Structure

My Official Outline

1. Introduction (2 pages)
    - 1.2 Research and Education Plan
    - 1.2 Broader Impacts

2 Background (1.75 pages)
    - 2.1 Limitations of Writing a Model
    - 2.2 Limitations of Exploring and Correcting a Model

3 Research Thrusts (9.25 pages)
    - 3.1 Core Techniques
        - 3.1.1 Formula Completion Suggestions — *Preliminary Work*
        - 3.1.2 Enumeration View
        - 3.1.3 Focus View
    - 3.2 Bidirectional Transformation Techniques — *Relevant Prior Work*
        - 3.2.1 Output Directed Debugging
        - 3.2.2 Solution-Driven Base Models — *Preliminary Work*
    - 3.3 New Applications
        - 3.3.1 Lynx: Live Programming IDE for Pardinus
        - 3.3.2 Modeling-ABCs: Interactive Educational Tool for Mathematical Logic

4 Related Work (.75 pages)

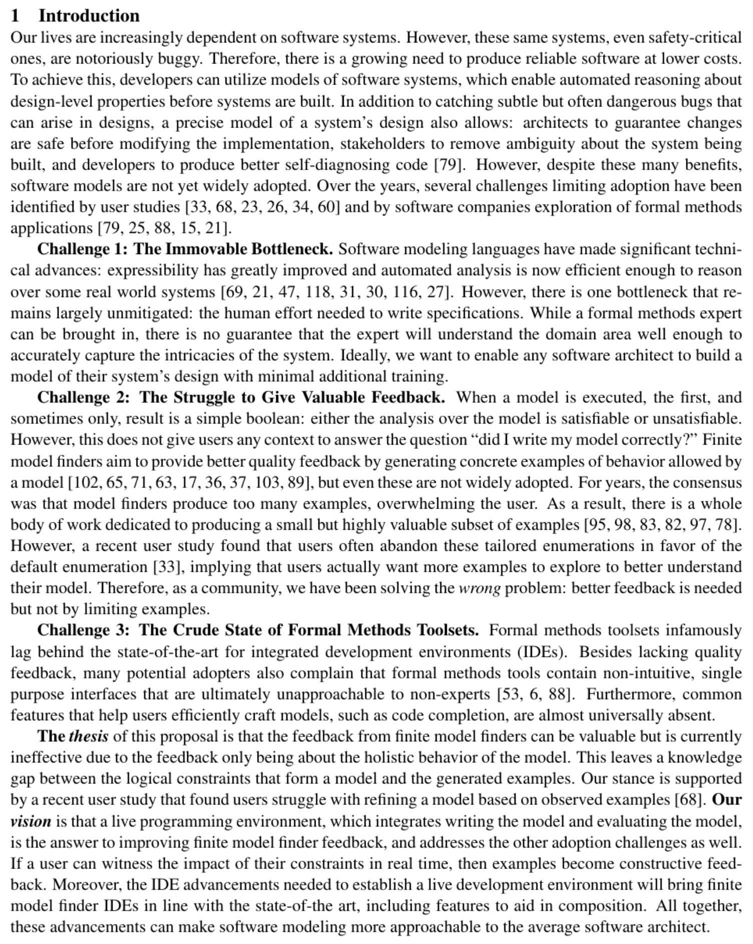
5 Education Plan (1 page)

6 Project Plan (.25 pages)

7 Results from Prior NSF Support (.25 pages)

Proposal Structure

- **The first page: sell the whole story**



1 Introduction

Our lives are increasingly dependent on software systems. However, these same systems, even safety-critical ones, are notoriously buggy. Therefore, there is a growing need to produce reliable software at lower costs. To achieve this, developers can utilize models of software systems, which enable automated reasoning about design-level properties before systems are built. In addition to catching subtle but often dangerous bugs that can arise in designs, a precise model of a system's design also allows: architects to guarantee changes are safe before modifying the implementation, stakeholders to remove ambiguity about the system being built, and developers to produce better self-diagnosing code [79]. However, despite these many benefits, software models are not yet widely adopted. Over the years, several challenges limiting adoption have been identified by user studies [33, 68, 23, 26, 34, 60] and by software companies exploration of formal methods applications [79, 25, 88, 15, 21].

Challenge 1: The Immovable Bottleneck. Software modeling languages have made significant technical advances: expressibility has greatly improved and automated analysis is now efficient enough to reason over some real world systems [69, 21, 47, 118, 31, 30, 116, 27]. However, there is one bottleneck that remains largely unmitigated: the human effort needed to write specifications. While a formal methods expert can be brought in, there is no guarantee that the expert will understand the domain area well enough to accurately capture the intricacies of the system. Ideally, we want to enable any software architect to build a model of their system's design with minimal additional training.

Challenge 2: The Struggle to Give Valuable Feedback. When a model is executed, the first, and sometimes only, result is a simple boolean: either the analysis over the model is satisfiable or unsatisfiable. However, this does not give users any context to answer the question "did I write my model correctly?" Finite model finders aim to provide better quality feedback by generating concrete examples of behavior allowed by a model [102, 65, 71, 63, 17, 36, 37, 103, 89], but even these are not widely adopted. For years, the consensus was that model finders produce too many examples, overwhelming the user. As a result, there is a whole body of work dedicated to producing a small but highly valuable subset of examples [95, 98, 83, 82, 97, 78]. However, a recent user study found that users often abandon these tailored enumerations in favor of the default enumeration [33], implying that users actually want more examples to explore to better understand their model. Therefore, as a community, we have been solving the *wrong* problem: better feedback is needed but not by limiting examples.

Challenge 3: The Crude State of Formal Methods Toolsets. Formal methods toolsets infamously lag behind the state-of-the-art for integrated development environments (IDEs). Besides lacking quality feedback, many potential adopters also complain that formal methods tools contain non-intuitive, single purpose interfaces that are ultimately unapproachable to non-experts [53, 6, 88]. Furthermore, common features that help users efficiently craft models, such as code completion, are almost universally absent.

The *thesis* of this proposal is that the feedback from finite model finders can be valuable but is currently ineffective due to the feedback only being about the holistic behavior of the model. This leaves a knowledge gap between the logical constraints that form a model and the generated examples. Our stance is supported by a recent user study that found users struggle with refining a model based on observed examples [68]. ***Our vision*** is that a live programming environment, which integrates writing the model and evaluating the model, is the answer to improving finite model finder feedback, and addresses the other adoption challenges as well. If a user can witness the impact of their constraints in real time, then examples become constructive feedback. Moreover, the IDE advancements needed to establish a live development environment will bring finite model finder IDEs in line with the state-of-the art, including features to aid in composition. All together, these advancements can make software modeling more approachable to the average software architect.

Big picture – why the problem space matters

Software correctness → modeling → adoption

What are the challenges this proposal will try to solve?

- Give supporting evidence
- Limitations of current work to address these
- Will be themes throughout the proposal

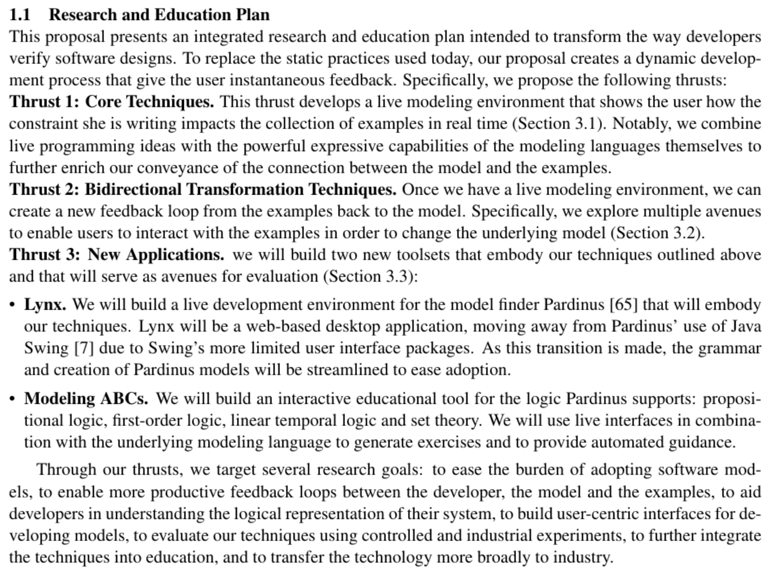
Thesis/Vision statement

Thesis: What is key insight I will design my solutions around

Vision: What is my solution and how does it tackle the challenges

Proposal Structure

- **The second page: get into the details**



1.1 Research and Education Plan

This proposal presents an integrated research and education plan intended to transform the way developers verify software designs. To replace the static practices used today, our proposal creates a dynamic development process that give the user instantaneous feedback. Specifically, we propose the following thrusts:

Thrust 1: Core Techniques. This thrust develops a live modeling environment that shows the user how the constraint she is writing impacts the collection of examples in real time (Section 3.1). Notably, we combine live programming ideas with the powerful expressive capabilities of the modeling languages themselves to further enrich our conveyance of the connection between the model and the examples.

Thrust 2: Bidirectional Transformation Techniques. Once we have a live modeling environment, we can create a new feedback loop from the examples back to the model. Specifically, we explore multiple avenues to enable users to interact with the examples in order to change the underlying model (Section 3.2).

Thrust 3: New Applications. we will build two new toolsets that embody our techniques outlined above and that will serve as avenues for evaluation (Section 3.3):

- **Lynx.** We will build a live development environment for the model finder Pardinus [65] that will embody our techniques. Lynx will be a web-based desktop application, moving away from Pardinus' use of Java Swing [7] due to Swing's more limited user interface packages. As this transition is made, the grammar and creation of Pardinus models will be streamlined to ease adoption.
- **Modeling ABCs.** We will build an interactive educational tool for the logic Pardinus supports: propositional logic, first-order logic, linear temporal logic and set theory. We will use live interfaces in combination with the underlying modeling language to generate exercises and to provide automated guidance.

Through our thrusts, we target several research goals: to ease the burden of adopting software models, to enable more productive feedback loops between the developer, the model and the examples, to aid developers in understanding the logical representation of their system, to build user-centric interfaces for developing models, to evaluate our techniques using controlled and industrial experiments, to further integrate the techniques into education, and to transfer the technology more broadly to industry.

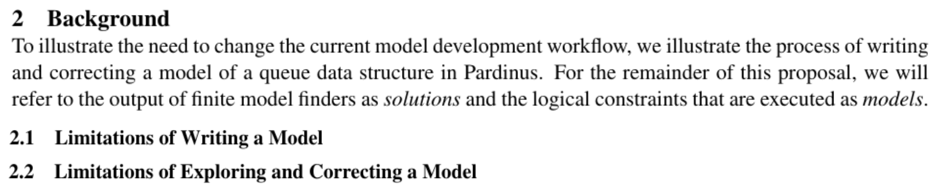
Overview of 3 research thrusts
- Elevator pitch for each thrust
- Highlight key novelty/insights
- Priming the reviewer's impression

Integration of research and education

Concluding list to reinforce all the outcomes of the research

Examples, Examples, Examples

- **Background – 1.75 pages**
  - Set up a running example to contextualize my proposed work later
  - Highlight the challenges from page 1 to reinforce the motivation



2 Background

To illustrate the need to change the current model development workflow, we illustrate the process of writing and correcting a model of a queue data structure in Pardinus. For the remainder of this proposal, we will refer to the output of finite model finders as *solutions* and the logical constraints that are executed as *models*.

2.1 Limitations of Writing a Model

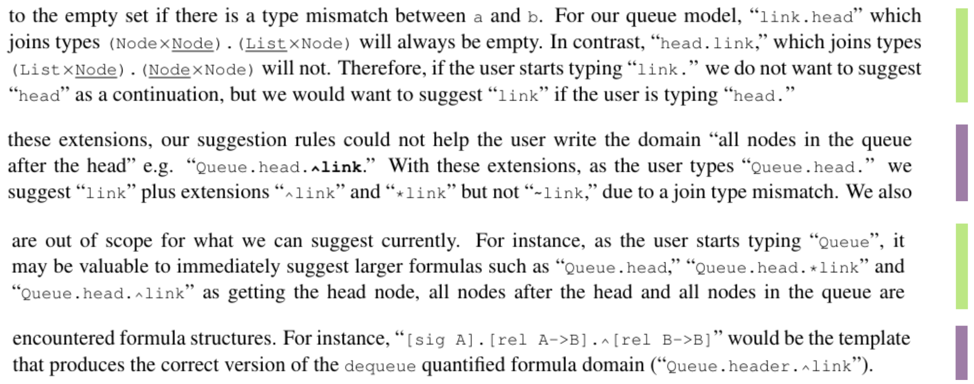
2.2 Limitations of Exploring and Correcting a Model

Balance:

- Simple enough to illustrate the issues
- Some complexity to make the research not look trivial

Examples, Examples, Examples

- **Let your examples do the work:**
  - Research content is dense
  - Helps reviewers
  - Helps you draft your ideas and think through potential pitfalls



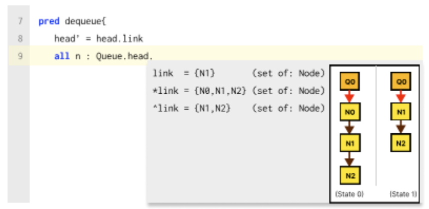
to the empty set if there is a type mismatch between `a` and `b`. For our queue model, "`link.head`" which joins types `(Node×Node).(List×Node)` will always be empty. In contrast, "`head.link`," which joins types `(List×Node).(Node×Node)` will not. Therefore, if the user starts typing "`link.`" we do not want to suggest "`head`" as a continuation, but we would want to suggest "`link`" if the user is typing "`head.`"

these extensions, our suggestion rules could not help the user write the domain "all nodes in the queue after the head" e.g. "`Queue.head.^link`." With these extensions, as the user types "`Queue.head.`" we suggest "`link`" plus extensions "`^link`" and "`*link`" but not "`~link`," due to a join type mismatch. We also

are out of scope for what we can suggest currently. For instance, as the user starts typing "`Queue`", it may be valuable to immediately suggest larger formulas such as "`Queue.head`," "`Queue.head.*link`" and "`Queue.head.^link`" as getting the head node, all nodes after the head and all nodes in the queue are

encountered formula structures. For instance, "`[sig A].[rel A->B].^[rel B->B]`" would be the template that produces the correct version of the `dequeue` quantified formula domain ("`Queue.header.^link`").

Figure 3: Lynx Wire-Frame of Suggestion Box

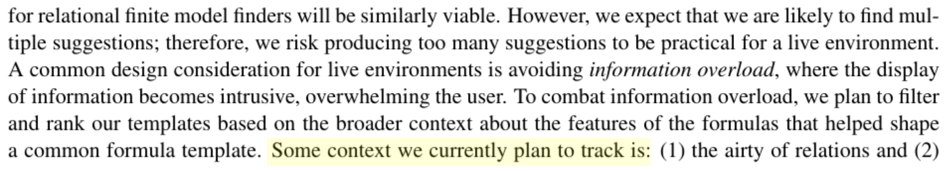


```
pred dequeue{
    head' = head.link
    all n : Queue.head.
                     link  = {N1}        (set of: Node)
                     *link = {N0,N1,N2}  (set of: Node)
                     ^link = {N1,N2}     (set of: Node)
```

Examples used in just 3.1.1 Formula Completion Suggestions (*preliminary and proposed work*)

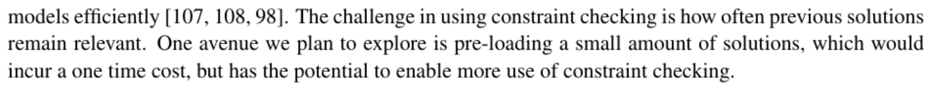
Acknowledge Challenges and Alternatives

- **Research does not always work out – and that's ok – and its also ok to be up front about that in NSF proposals**
    - What are your backup plans for different parts of the proposal?
    - If you are aiming for a tradeoff, what if it does not work out?



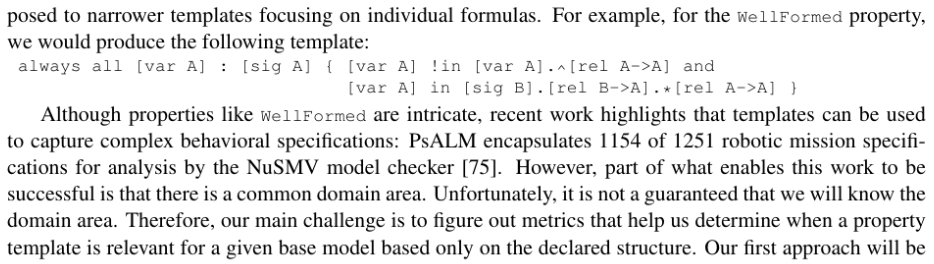
for relational finite model finders will be similarly viable. However, we expect that we are likely to find multiple suggestions; therefore, we risk producing too many suggestions to be practical for a live environment. A common design consideration for live environments is avoiding *information overload*, where the display of information becomes intrusive, overwhelming the user. To combat information overload, we plan to filter and rank our templates based on the broader context about the features of the formulas that helped shape a common formula template. Some context we currently plan to track is: (1) the airty of relations and (2)

Outline gameplans, even if broad, for potential issues and include these in the evaluation plan



models efficiently [107, 108, 98]. The challenge in using constraint checking is how often previous solutions remain relevant. One avenue we plan to explore is pre-loading a small amount of solutions, which would incur a one time cost, but has the potential to enable more use of constraint checking.

Address limitations that arise due to compromises for [scalability]



posed to narrower templates focusing on individual formulas. For example, for the `WellFormed` property, we would produce the following template:

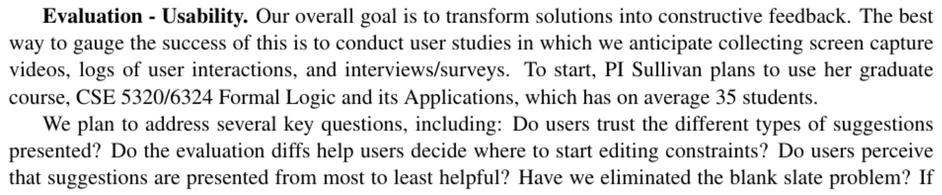
```
always all [var A] : [sig A] { [var A] !in [var A].^[rel A->A] and
                               [var A] in [sig B].[rel B->A].*[rel A->A] }
```

Although properties like `WellFormed` are intricate, recent work highlights that templates can be used to capture complex behavioral specifications: PsALM encapsulates 1154 of 1251 robotic mission specifications for analysis by the NuSMV model checker [75]. However, part of what enables this work to be successful is that there is a common domain area. Unfortunately, it is not a guaranteed that we will know the domain area. Therefore, our main challenge is to figure out metrics that help us determine when a property template is relevant for a given base model based only on the declared structure. Our first approach will be

Cite relevant examples where something similar has been successful – and a roadmap for how it might port to your work and what new problems you'll address

Evaluations

- **Embedded in Thrust 3 for me**
  - One tool evaluates thrusts 1 and 2
  - One tool targets educational efforts based on thrusts 1 and 2
  - Prior funded grants, "evaluation is vague" would be in my reviews
    - Examples of past grants not always perfect
  - Changed up my evaluation style: not a weakness called out in my reviews this time



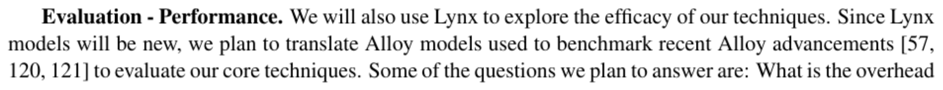
Evaluation - Usability. Our overall goal is to transform solutions into constructive feedback. The best way to gauge the success of this is to conduct user studies in which we anticipate collecting screen capture videos, logs of user interactions, and interviews/surveys. To start, PI Sullivan plans to use her graduate course, CSE 5320/6324 Formal Logic and its Applications, which has on average 35 students.

We plan to address several key questions, including: Do users trust the different types of suggestions presented? Do the evaluation diffs help users decide where to start editing constraints? Do users perceive that suggestions are presented from most to least helpful? Have we eliminated the blank slate problem? If

List target user groups

List what you will collect to perform the evaluation

List direct questions you want to answer



Evaluation - Performance. We will also use Lynx to explore the efficacy of our techniques. Since Lynx models will be new, we plan to translate Alloy models used to benchmark recent Alloy advancements [57, 120, 121] to evaluate our core techniques. Some of the questions we plan to answer are: What is the overhead

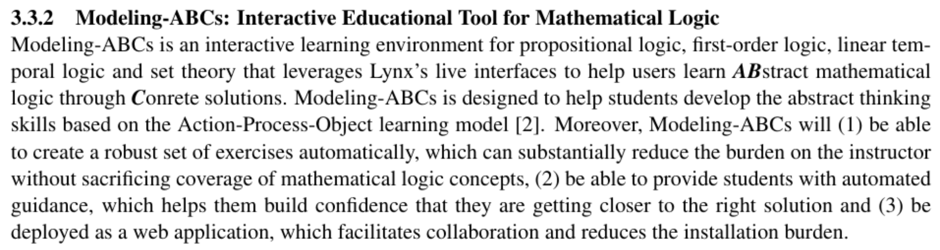
List where evaluation material will come from

Education Connection

- How would I utilize this research in a classroom?
- How can I enable others to utilize this research in the classroom?
- How could this research help with outreach?

My Approach: *Modeling ABCs* – a tool embodying the advancements of thrust 1 and 2 but designed specifically for education down to UIs

- Exercise Concepts
- Exercise Generation
- Automated Guidance
- Evaluation



3.3.2 Modeling-ABCs: Interactive Educational Tool for Mathematical Logic

Modeling-ABCs is an interactive learning environment for propositional logic, first-order logic, linear temporal logic and set theory that leverages Lynx's live interfaces to help users learn ***AB***stract mathematical logic through ***C***onrete solutions. Modeling-ABCs is designed to help students develop the abstract thinking skills based on the Action-Process-Object learning model [2]. Moreover, Modeling-ABCs will (1) be able to create a robust set of exercises automatically, which can substantially reduce the burden on the instructor without sacrificing coverage of mathematical logic concepts, (2) be able to provide students with automated guidance, which helps them build confidence that they are getting closer to the right solution and (3) be deployed as a web application, which facilitates collaboration and reduces the installation burden.

I looked into how people learn abstractions

What features would be helpful for professors to adopt the tool?

What features would be helpful to students compared to current tools?

How to deploy?

Education Connection

My Education Plan Outline

Undergraduate Course Development

- Using Modeling ABCs

Interdisciplinary Graduate Course Development

- Using Modeling ABCs
- Looking beyond just CSE (math, philosophy, public policy)

Encouraging Underrepresented and Undergraduate Students

- Step to create a supported environment: On departments BPC committee, SWE faculty advisor
- Highlight past/planned mentorship strategies and/or opportunities (some universities support REUs for instance)

Graduate Student Advisement

- How you are helping them develop as researchers
- How you will prepare students for careers

K-12 Outreach

- Consider existing events that you could help bring to your area
- e.g. Girl Day during E-Week

Outcome Assessment

Things I Found Useful

- **Examples of past grants**
  - Past successful CAREERs from others
  - Past successful NSF proposals in general
- **Serving on a NSF Panel**
  - Great exposure to how reviewers think
- **This workshop over university generic workshop**
  - My university hosts a yearly CAREER/Young Investigator workshop but it covers all of the college of engineering departments
  - Expectations for these grants are not aligned across fields

Writing a Grant with a Baby

- **Proofreading, proofreading, proofreading**
  - Why read "llama llama red pajama?" when you could read your grant out loud
  - When sleep deprived, your writing gets weird
  - Plan buffer time for this – take breaks and come back to it
- **Examples**
  - Helped me focus and think things through when thoughts were jumbled
- **Coffee**
- **I had an amazing support system**
  - My husband stepped up when I needed time – including letting me sleep
  - My parents and his parents live in the same city as us - huge