
University of
Southampton

University of Southampton

Project Allocation with Event-B and ProB

Son Hoang, Abdolbaghi Rezazadeh, and Michael Butler
(University of Southampton, UK)

Rodin Workshop 2025 (10/06/2025)

Context

University of Southampton

- Annual allocation of students (Year 3, MSc) to supervisors.
- Challenges:
  - Growing student number
  - Multiple student per staff
  - Matching students to relevant projects
  - Staff loading constraints
  - Load balancing

Student's View of the Process

University of Southampton

- Select preferences from a list of topics
  - Up to 12 choices
  - Choices from different supervisors
- Allocated to a supervisor
- Most of students got a supervisor from their chosen list.

Issue

- Allocation software is no longer maintained
- Performance detoriated when
  - New programmes are added
  - Cohort's size increases
- Manual allocation required

Aims

University of Southampton

Formal Model

- Precisely specify the algorithm
- “Executable” for allocation
- Adaptable to future changes in the allocation process.
- (Not a focus): Proving the consistency of the algorithm

Requirements for Project Allocation

The assumptions (1/2)

University of Southampton

ASM 1	There is a finite set of programmes
ASM 2	There is a finite set of students
ASM 3	There is a finite set of staff

Requirements for Project Allocation

The assumptions (2/2)

University of Southampton

ASM 4	Each student is associated with a programme
ASM 5	Each staff is associated with a set of programmes
ASM 6	Students have a preference ranking (without duplication) of the supervisors
ASM 7	Each staff has a maximum number of students that they can supervise

Requirements for Project Allocation

The requirements

University of Southampton

REQ 8	A successful allocation must ensure that every student is allocated to a supervisor
REQ 9	A student's programme must match one of the supervisor's indicatedprogrammes
REQ 10	If a student has some preferences, then the allocated supervisor must be on their preference list

The Formal Context

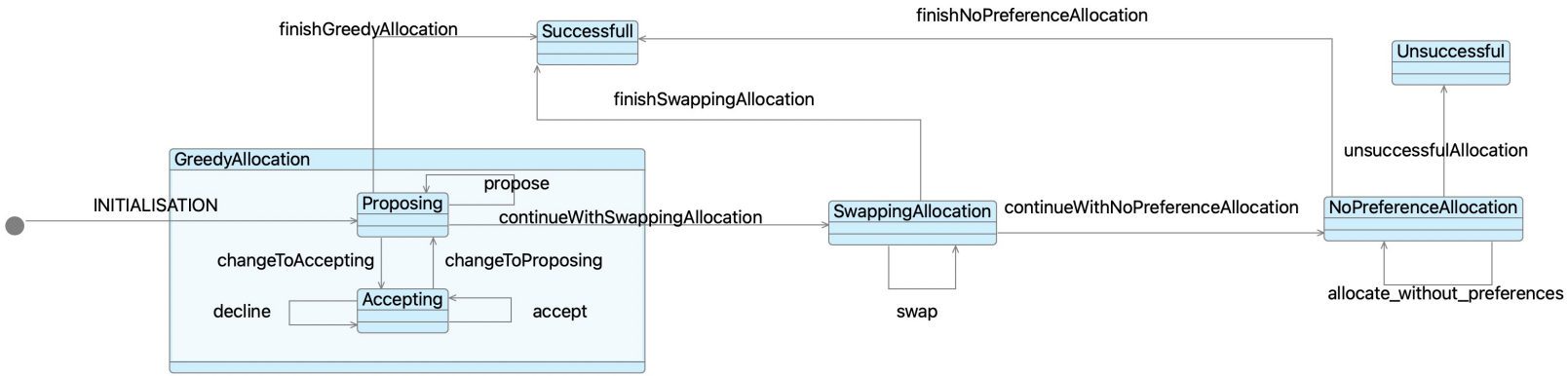
University of Southampton

```
1  sets
2    PROGRAMME STAFF STUDENT
3  constants
4    student_programme
5    staff_programmes
6    student_preferences
7    staff_limit
8  axioms
9    @axm1: finite(PROGRAMME) // ASM1
10   @axm2: finite(STUDENT) // ASM2
11   @axm3: finite(STAFF) // ASM3
12   @axm4: student_programme ∈ STUDENT→PROGRAMME // ASM4
13   @axm5: staff_programmes ∈ STAFF ↔ PROGRAMME
14   @axm6: dom(staff _programmes)=STAFF // ASM5
15   @axm7: student_preferences ∈ STUDENT→(STAFF→7 N) // ASM6
16   @axm8: staff_limit ∈ STAFF→N // ASM7
```

One-Shot Specification

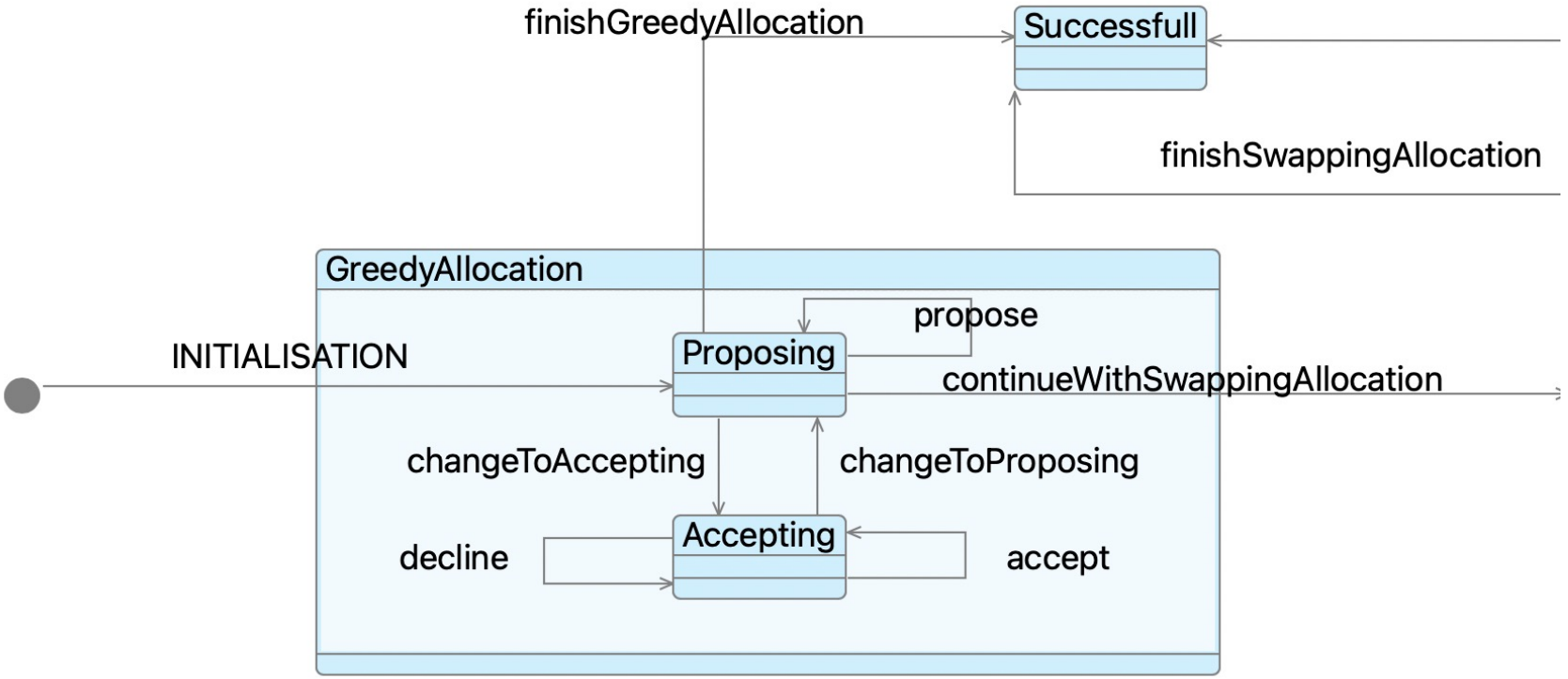
University of Southampton

```
1   event allocate
2   any allocation
3   where
4     @grd1: allocation ∈ STUDENT → STAFF // REQ8
5
6     @grd2: ∀student· student_programme(student)
7         ∈ staff_programmes[{allocation(student)}] // REQ9
8
9     @grd3: ∀student·student_preferences(student) ≠ ∅ ⇒
10        allocation(student) ∈ dom(student_preferences(student)) // REQ10
11
12    @grd4: ∀staff · card(allocation ▷ {staff}) ≤ staff_limit(staff) // REQ11
13  end
```



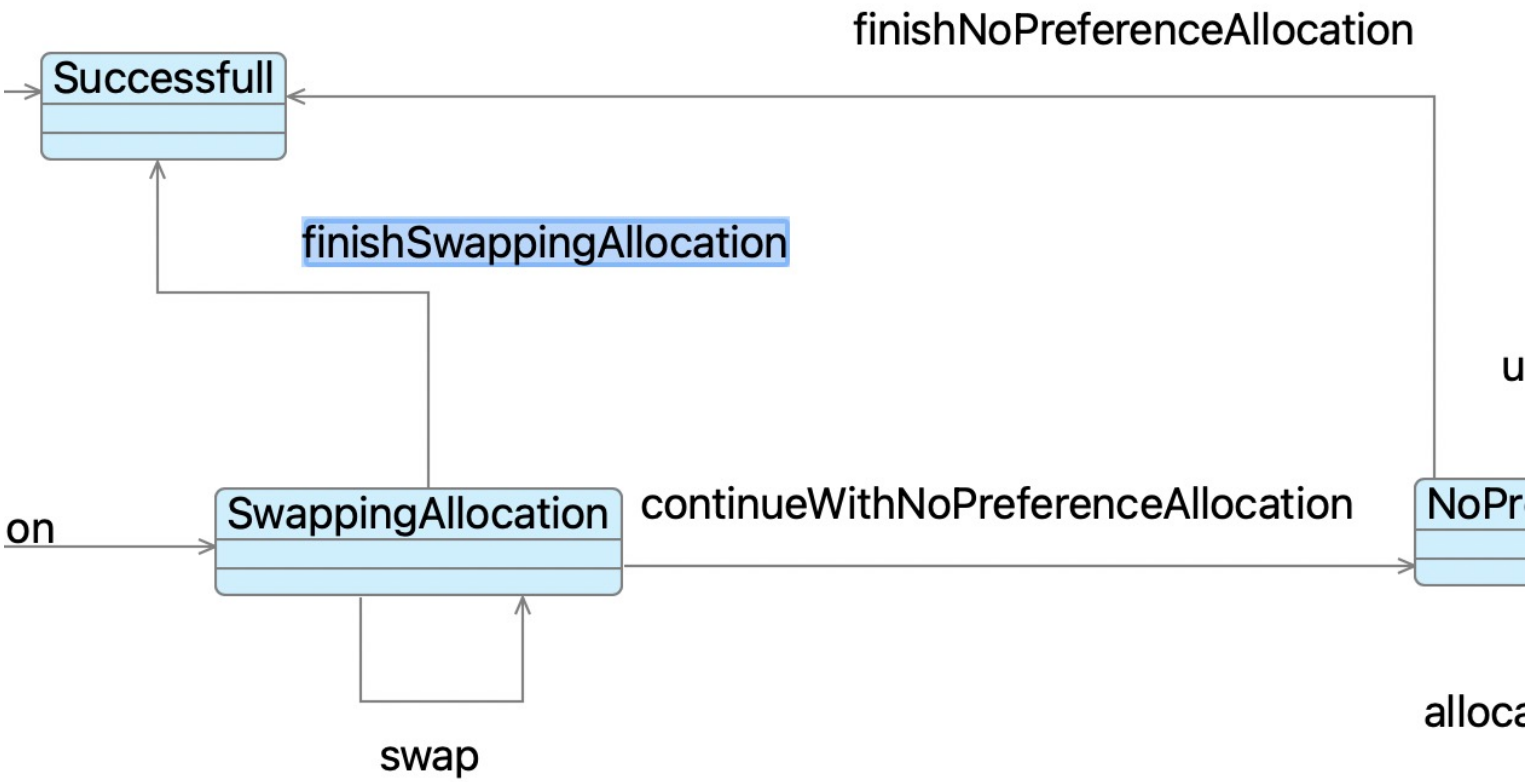
- Early stop if all students are allocated.
- Can be unsucessful
- No load balancing at the moment

Greedy Allocation Stage



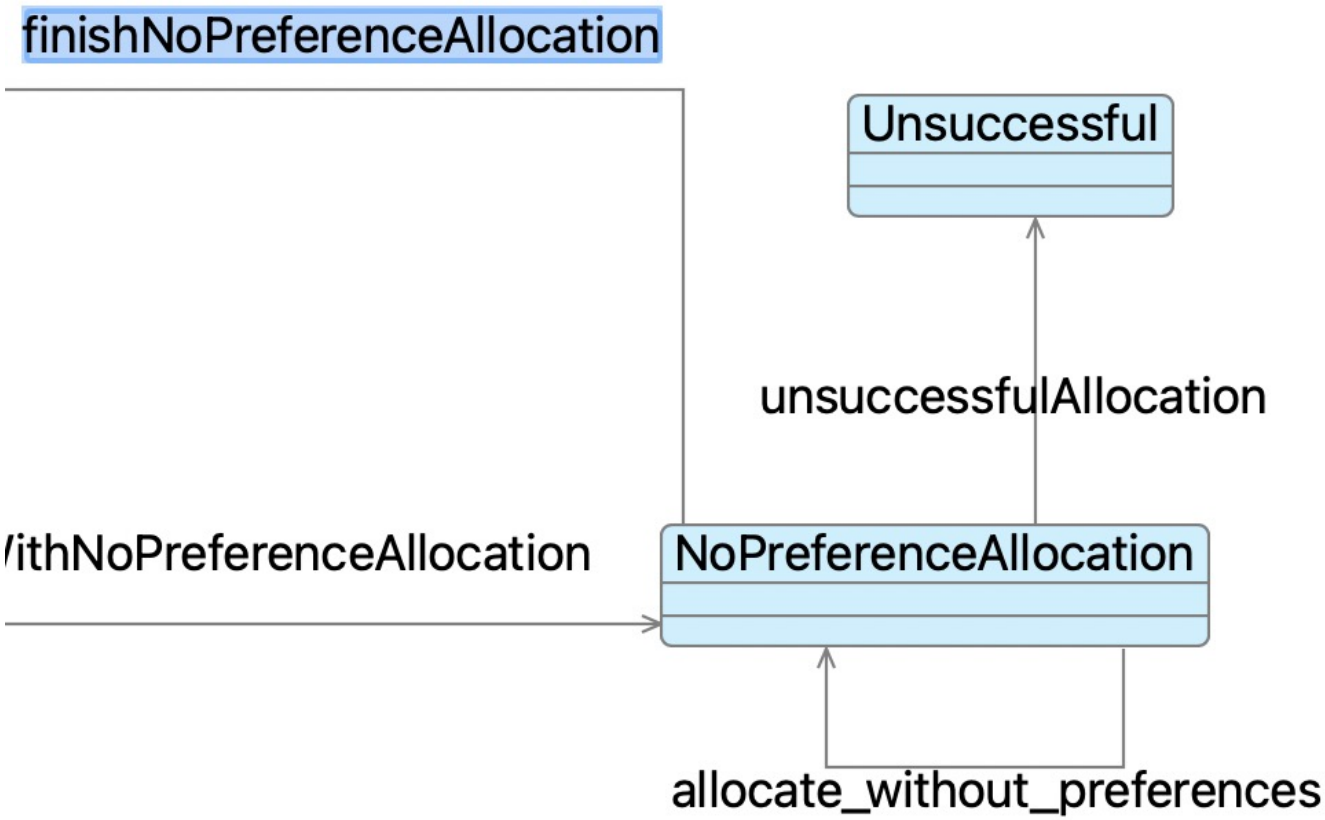
- Variant of “College Admission” algorithm (Gale and Shapley [1962]).
- Alternating between proposing and accepting
- Optimise the student preferences

Swapping



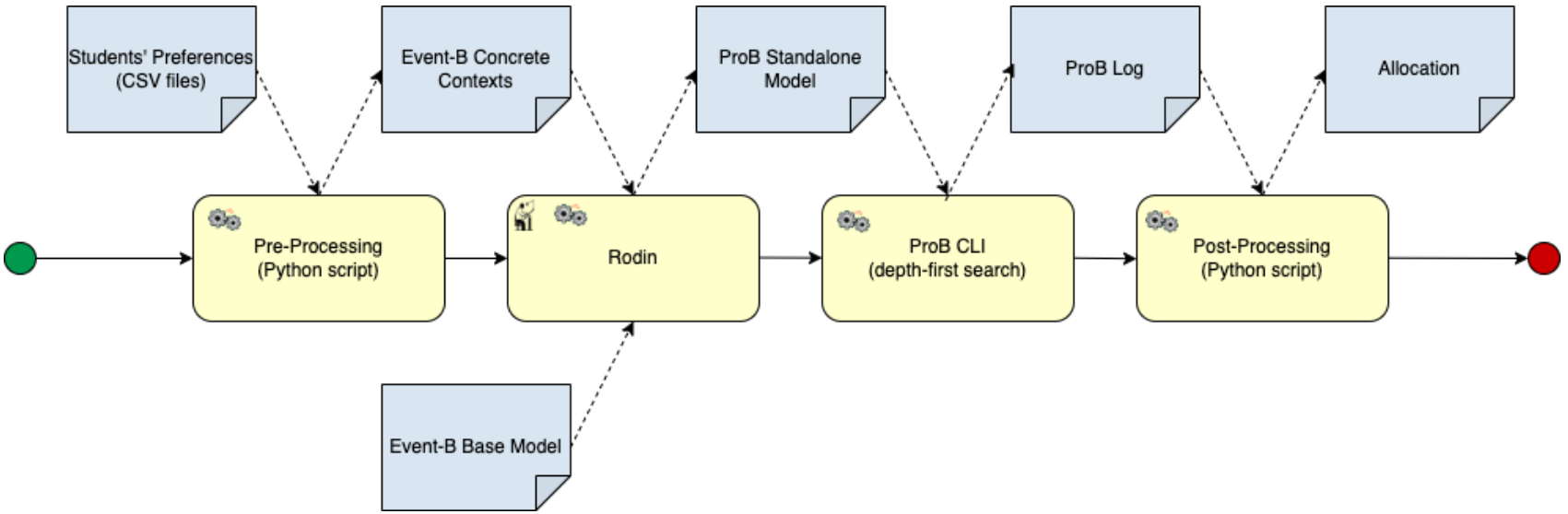
- Reallocate an allocated student to a different staff
- Allocate the supervisor to a new student
- Increase the number of allocated students.
- Trade off the student preferences.

No Preferences Allocation



- Allocated students without preferences to staff
- Increase the number of allocated students.

Tooling



- 244 MSc Students in 12 programmes
- 134 staff, each with a limit of 3 students
- 236 students allocated in Greedy Allocation Stage
- 5 students allocated in Swapping Allocation Stage
- 3 students allocated in No Preferences Allocation Stage
- Project Allocation is done 2 weeks early (compared to last year).

Conclusion

University of Southampton

Summary

- We can “execute” formal specification
- Bottle-neck: Rodin analyses the conrete context.

What’s Next

- By pass Rodin to generate ProB Standalone input directly.
- Add load balancing
- Update the algorithm to deal with topic choice
- Add properties (as invariants) for the algorithms
- Study other allocation problems/algorithms
- Define Language to specify algorithms (PlusCal-like for Event-B)

References I

David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1): 9–15, January 1962. URL http://www.jstor.org/stable/2312726?origin=JSTOR-pdf.