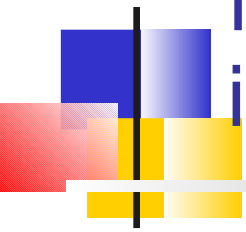


Runtime support for region-based memory management in Mercury

Quan Phan*, Zoltan Somogyi**, Gerda Janssens*

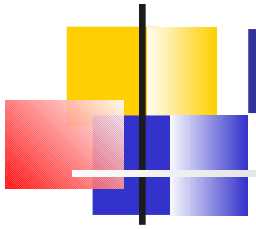
* DTAI, KU Leuven, Belgium.

** Computer Science and Software Engineering Department, University of Melbourne, Australia.

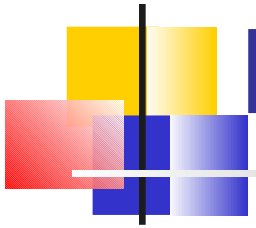
ISMM 2008

Tucson, Arizona, USA

Region-based Memory Management (RBMM)



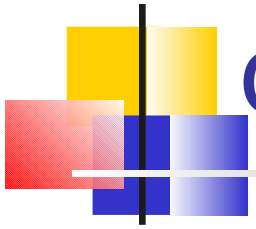
- Idea:
  - Group heap objects of the same lifetime into regions.
  - Reclaim garbage by destroying region as a whole.
- Advantages:
  - Small runtime overhead
    - No runtime detection of garbage
  - Often achieve good memory reuse.
  - Good chance of better data locality
    - Related data kept together.



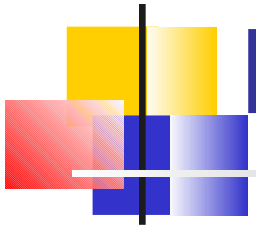
Mercury

- Logic/functional programming language
  - developed at Melbourne Univ.
  - declarative language aims at large-scale application development.
- Mercury's syntax is similar to Prolog's
- Explicit declarations
  - Types, modes, determinism.

Mercury's types, modes, determinism.

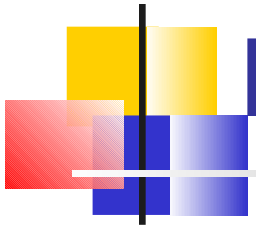


- Types: ~ Haskell's.
  - list(int) ---> [] ; [int | list(int)].
- Modes: instantiation of arguments of predicates.
  - in: ground → ground, out: free → ground.
  - a mode of a predicate: modes for its arguments → procedure.
- Determinism: # possible solutions of a procedure.



Mercury predicate

```
:- pred append(list(int), list(int), list(int)).
:- mode append(in, in, out) is det.
:- mode append(out, out, in) is multi.
append([], Y, Y).
append([Xe | Xs], Y, [Xe | Zs]) :-
   append(Xs, Y, Zs).
```

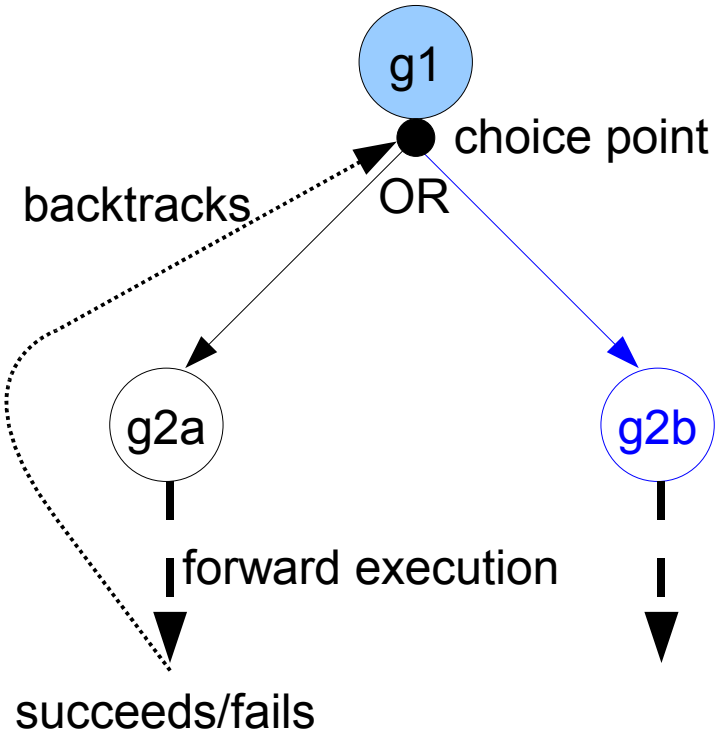


Backtracking

- Due to nondeterminism
  - Disjunction:
    - (g_1 ; ...; g_i ; ...; g_n)
    - Make a choice and backtracks into the disjunction later.
  - if g1 then g2 else g3
    - Semantically equivalent to the disjunction:
      - (g1, g2); (not g1, g3).
    - Try g1, if succeeds, execute g2. If fails, backtracks to g3 *as if* g1 had not been tried.

Backward execution

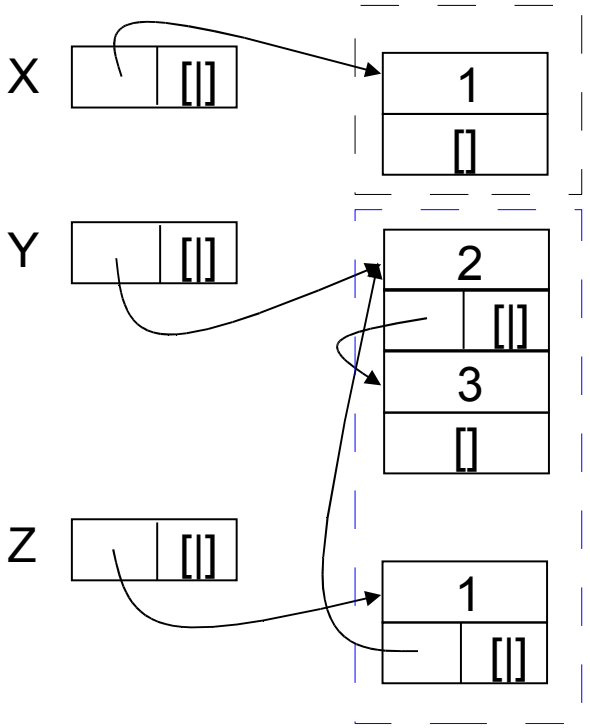
- ..., g1, (g2a; g2b), g3, ...
- Forward execution containing g2a.
- Backtrack to g2b: backward execution.
- Backward liveness: live during backward execution.

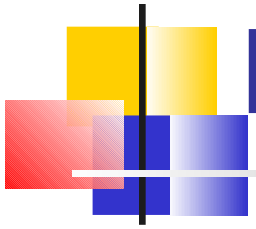


RBMM for Mercury

- Example: the call to append([1], [2, 3], Z) in the first mode.

```
% (in, in, out) is det.
append(X, Y, Z) :-
(
   X == [],
   Z := Y
;
   X => [Xe | Xs],
   append(Xs, Y, Zs),
   Z <= [Xe | Zs]
).
```

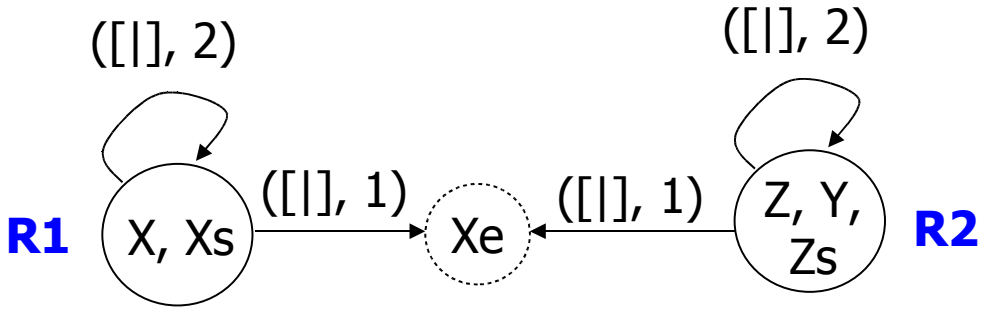




RBMM for Mercury

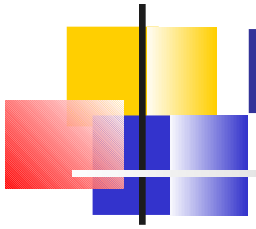
- Program analysis and transformation
  - Q. Phan and G. Janssens. *Static region analysis for Mercury.* ICLP 2007.
    - Regions.
    - Region liveness.
    - Mercury to region-annotated Mercury.
    - Often achieve good memory reuse.
  - S. Cherem and R. Rugina. *Region analysis and transformation for Java programs*. ISMM 2004.

Region-annotated Mercury




```
% (in, in, out) is det.
append(X, Y, Z) :-
(
   X == [],
   Z := Y
;
   X => [Xe | Xs],
   append(Xs, Y, Zs),
   Z <= [Xe | Zs]
).
```

```
% (in, in, out) is det.
append(X, Y, Z, R1, R2) :-
(
   X == [],
   remove(R1),
   Z := Y
;
   X => [Xe | Xs],
   append(Xs, Y, Zs, R1, R2),
   Z <= [Xe | Zs] in R2
).
```

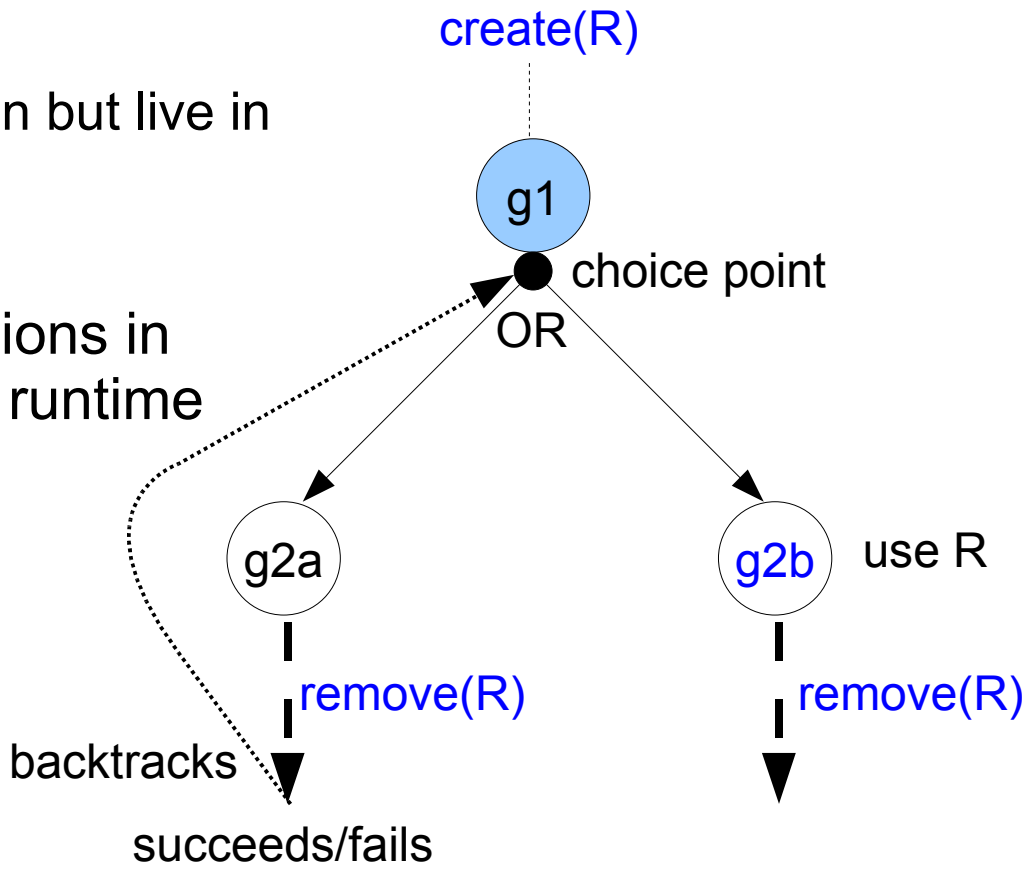


Runtime support

- Basic support
  - Regions, region instructions, allocation into regions.
  - Needed in any RBMM systems.
  - Mercury: only enough for programs with no backtracking.
- Support for backtracking
  - Liveness w.r.t forward execution.
  - Backtracking causes problems.
- How to support backtracking with little impact on deterministic code??
  - Less than 5% of Mercury code is nondeterministic.

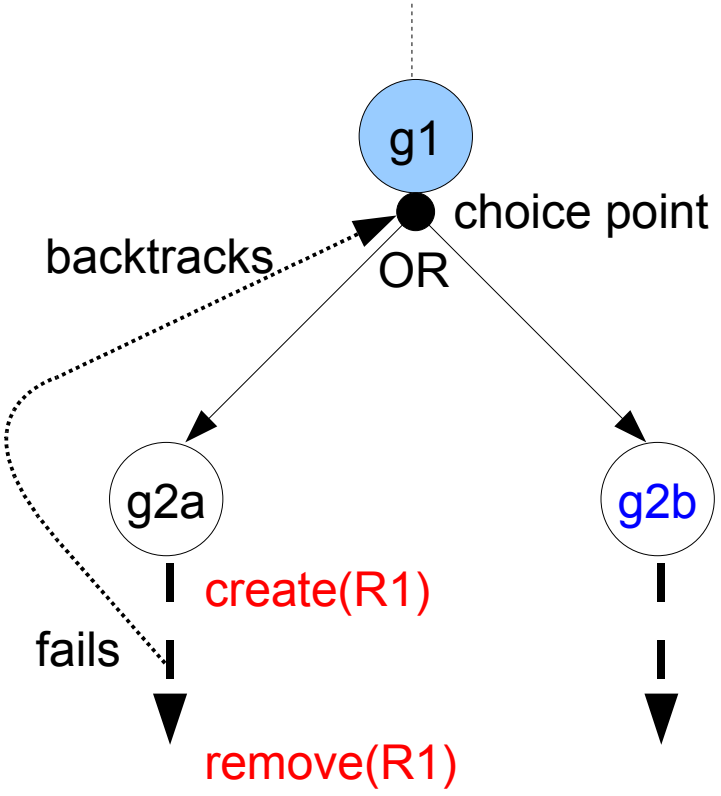
Impact of backtracking: Region resurrection.

- Region resurrection
  - Dead in forward execution but live in backward execution.
  - E.g., R.
- Destroy backward live regions in forward execution causes runtime errors.



Impact of backtracking: Instant reclaiming.

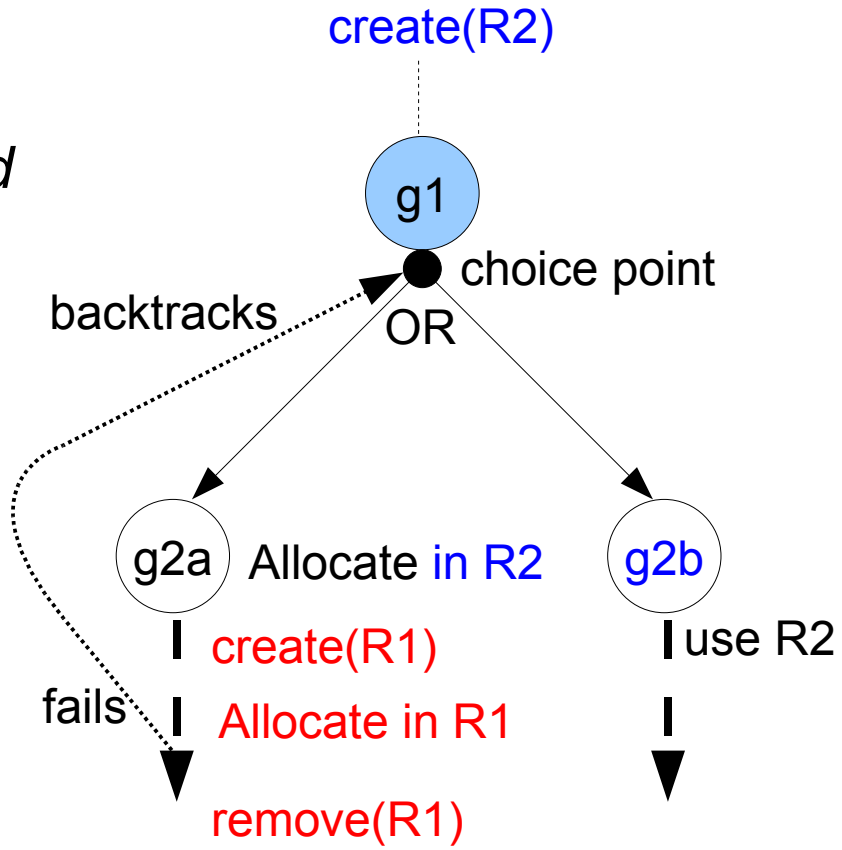
- Instant reclaiming
  - When backtracks to a choice point, allocations in the backtracked-over execution can be instantly reclaimed.
- Popularly used in logic programming systems.
- 1st case*: New* regions with respect to the choice point: R1
- Reclaim R1 before starting the backward execution containing g2b.



Impact of backtracking: Instant reclaiming.

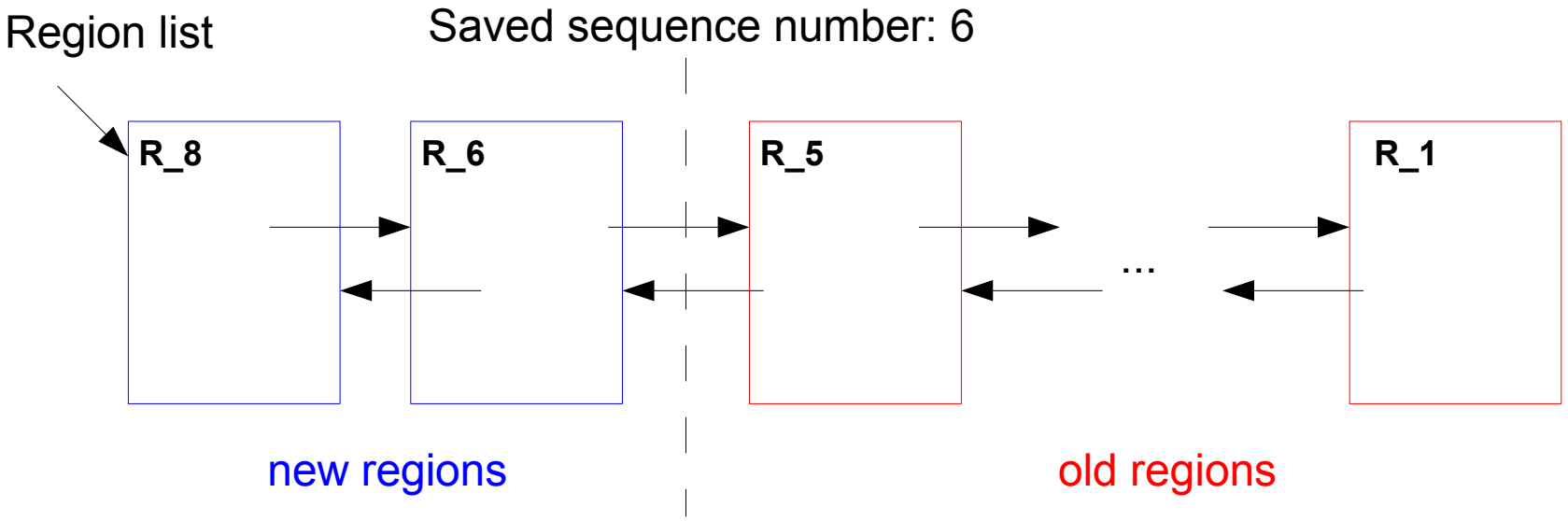
..., g1, (g2a ; g2b), g3...

- Instant reclaiming ...
- 2nd case: Allocations into *existing/old* regions: R2 (not R1).



Old vs. new regions, region list

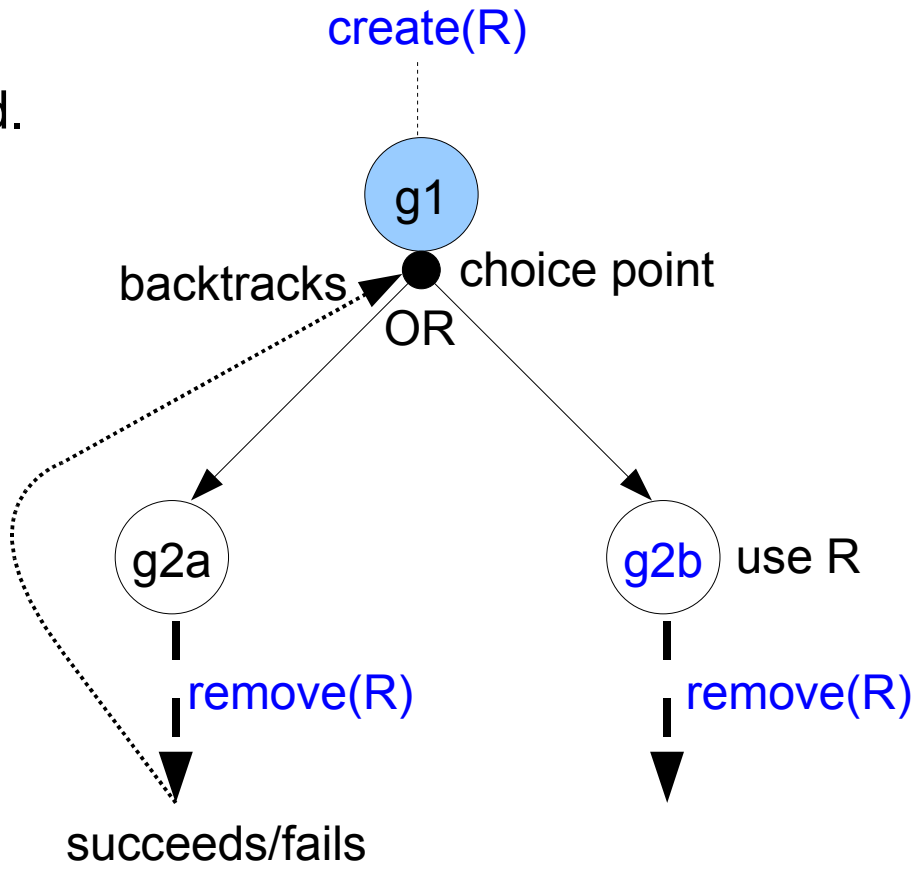
- Maintain a global region sequence number.



Support for nondet disjunction: Region resurrection.

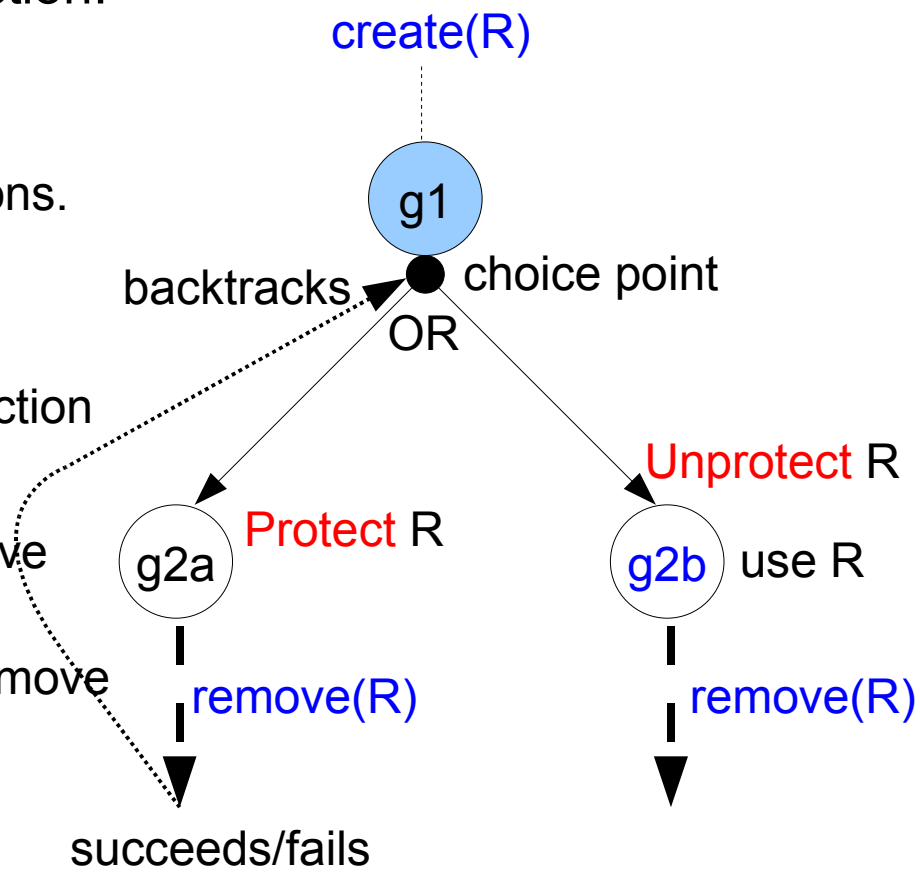
..., g1, (g2a ; g2b), g3, ...

- Nondet: any disjuncts may succeed.
  - Both g2a and g2b.
- Backtrack from *outside*
- → backward live regions ≈ all old regions: e.g., R.



Support for nondet disjunction: Region resurrection.

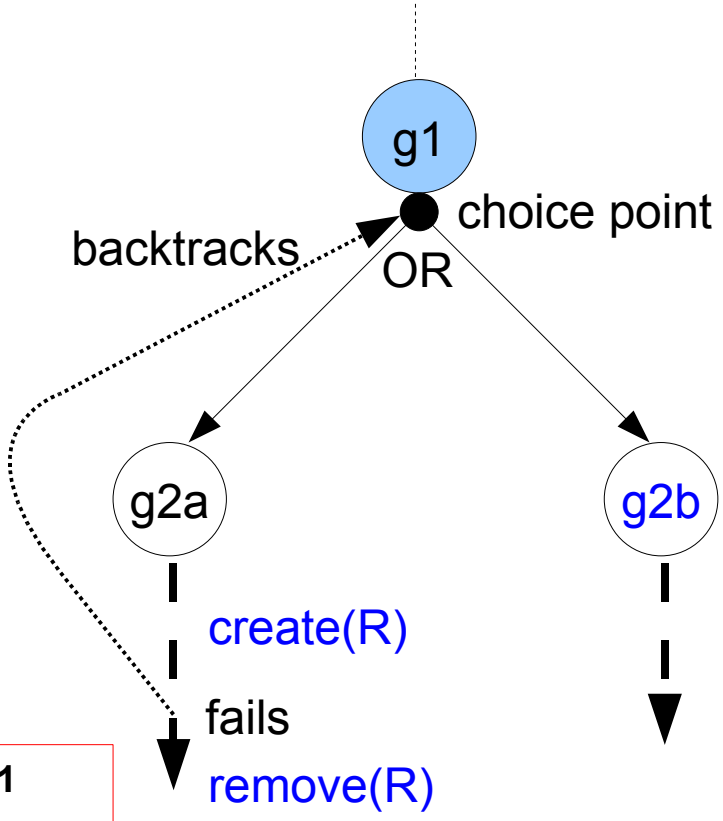
- Protect R at the entry to the disjunction: before g2a.
  - Save the global sequence number.
  - remove instruction: ignore *old* regions.
- Unprotect R at the start of the last disjunct: g2b.
  - No longer backtrack into the disjunction
  - Clear the saved number
  - remove instructions become effective again.
  - R is destroyed when the second remove is reached.

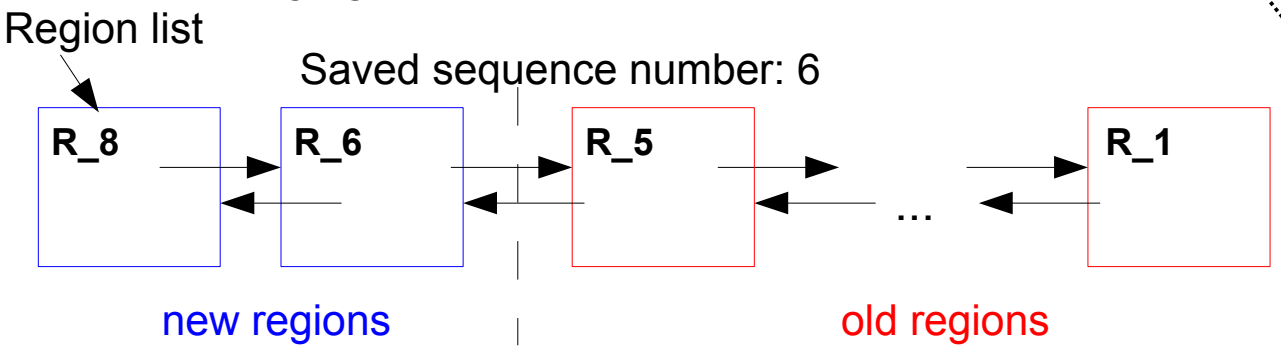


Support for nondet disjunction: Instant reclaiming

..., g1, (g2a ; g2b), g3, ...

- Instant reclaiming new regions
  - Already save the global sequence number.
  - When backtrack to a non-first disjunct: g2b
    - traverse the region list
    - reclaim regions until seeing an old one.

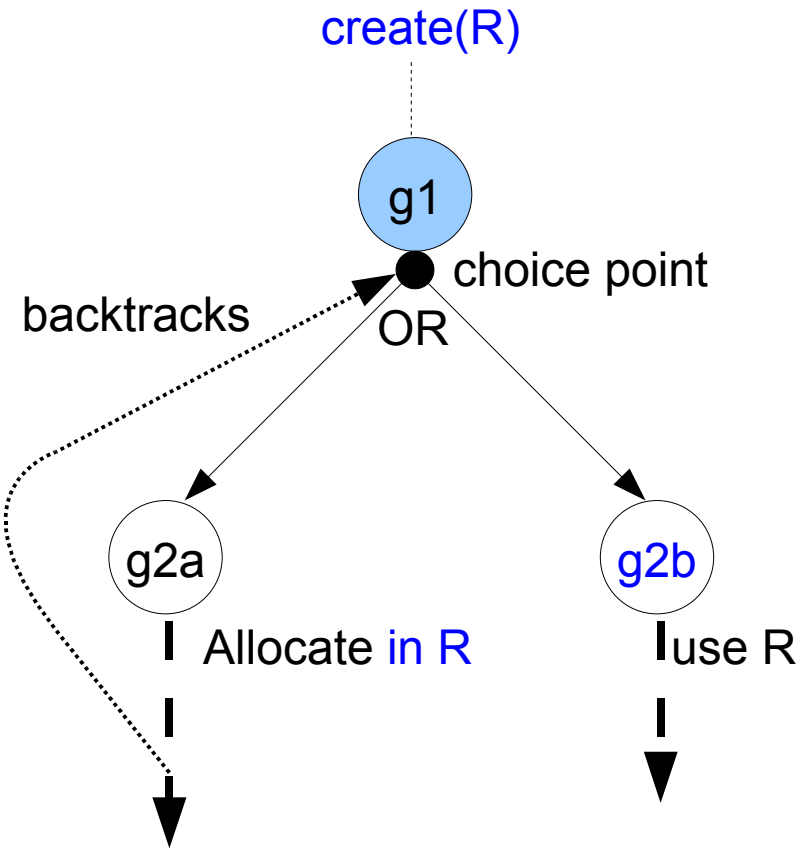




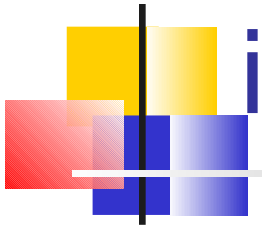
Support for nondet disjunction: Instant reclaiming

..., g1, (g2a ; g2b), ...

- Instant reclaiming new allocations
  - R is an old region.
  - Save the size of R at entry to the disjunction.
  - Instant reclaim by restoring at start of any non-first disjunct: g2b.

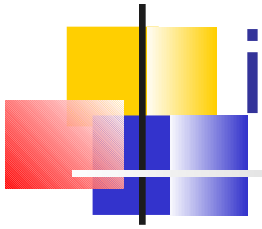


Optimized support for if-then-else



- If-then-else:
  - Efficient implementation.
  - Support if-then-else without damaging its efficiency.
- Similar support needed.
  - Region resurrection: protecting backward live regions.
  - Instant reclaiming at start of the else part.

Optimized support for if-then-else

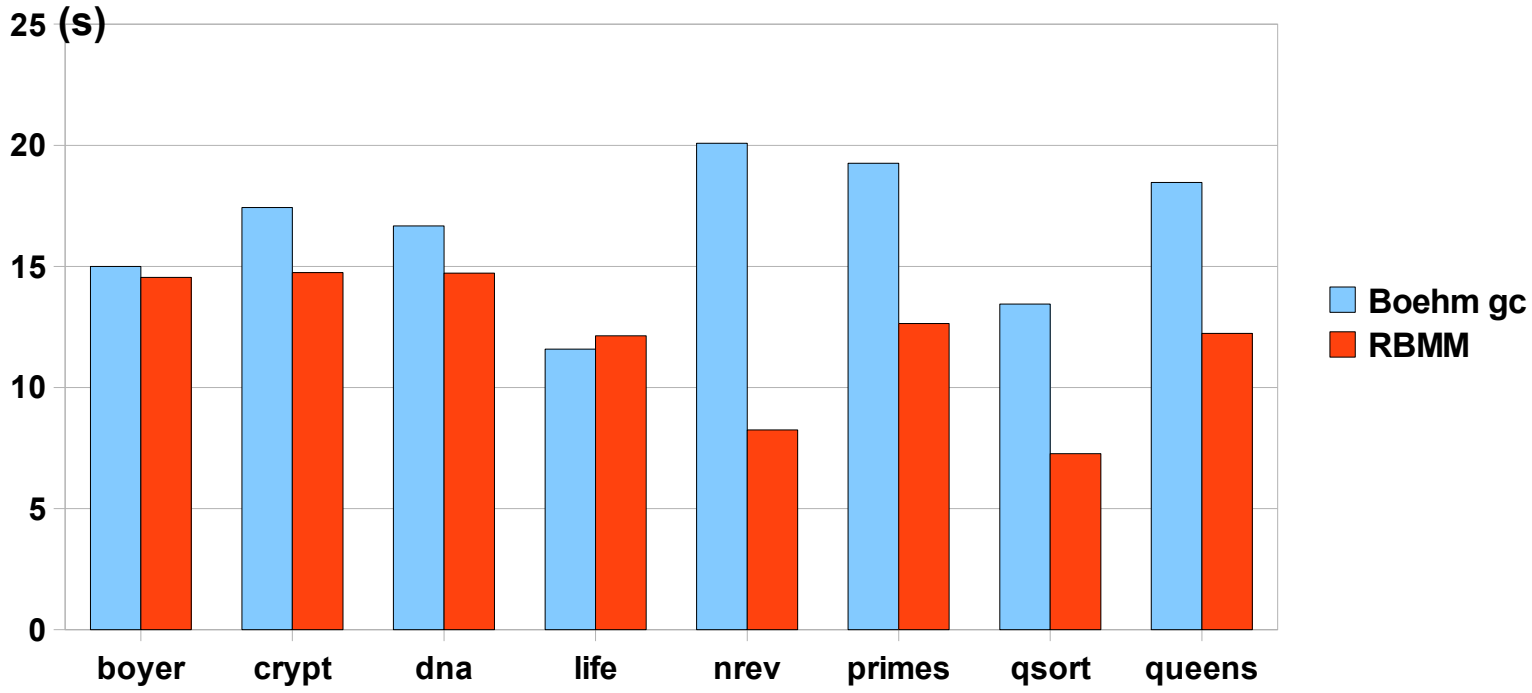


- Backtrack happens from inside the condition goal
  - Only support for changes in the condition
    - Protect backward live regions removed *in the condition*,
    - Instant reclaiming new regions created *in the condition*,
    - Instant reclaiming new allocations happen *in the condition*.
- These changes can be computed from region analysis information.
- No changes:
  - No support added.
  - Condition goals are often simple tests → maintaining efficiency.

Runtime performance

Mercury compiler that uses Boehm gc vs. Mercury compiler with RBMM.

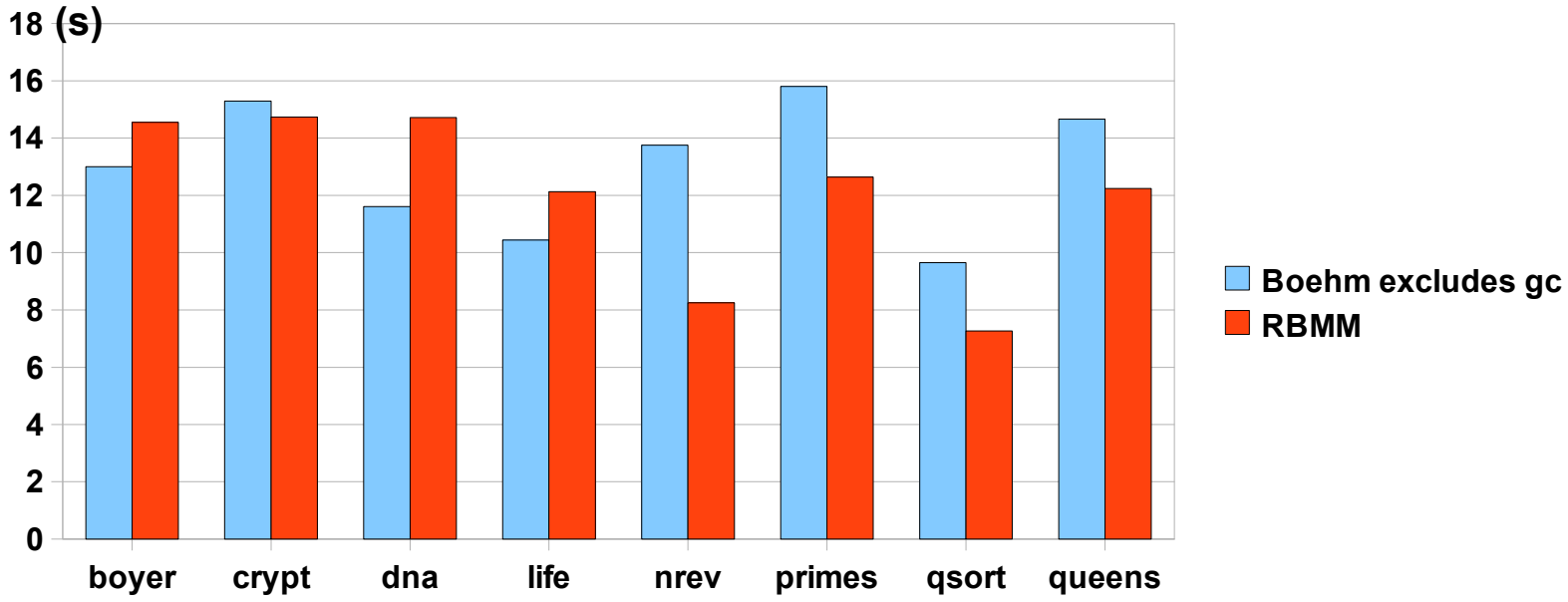
- Average speedup 25%.
- 2 nondet programs: crypt & queens.
- boyer and life: substantial cost of supporting backtracking.



Runtime performance

Exclude gc time:

- RBMM still better in 5 programs: better data locality → speedup due to better cache behaviour.

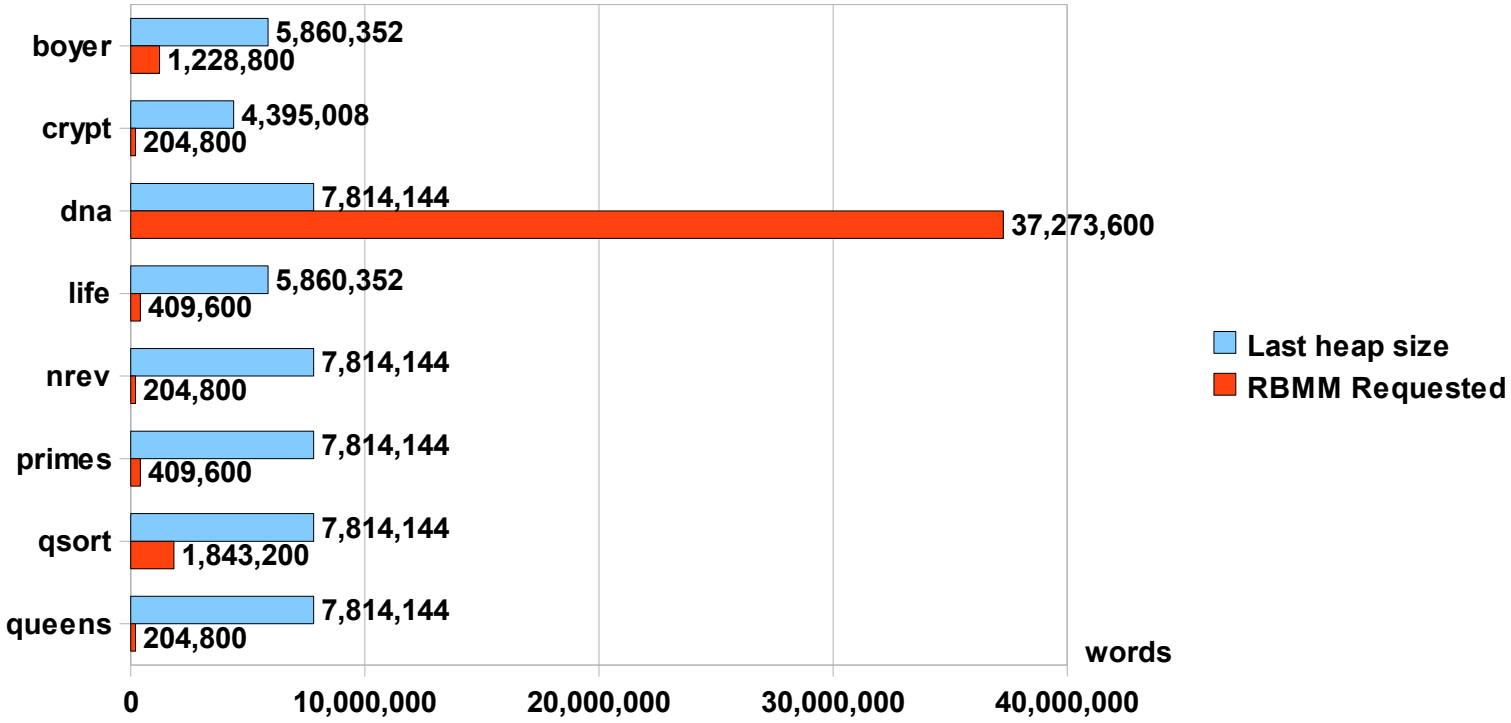


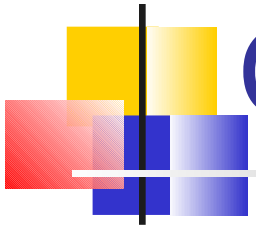
Memory consumption

Region page size 2k (words).

Initial RBMM size 200k, 200k/increase.

Initial heap size ~ 4M words (default in Mercury).





Conclusions

- Our results suggest that
  - RBMM can be implemented with modest runtime overhead.
  - Better data locality.
  - → overall speedup.
- Related work:
  - RBMM for Prolog: [K. Sagonas and H. Malkhom @ ICLP 2002]
    - Require different algorithms due to the significant difference between the two languages.
- Future work:
  - Modify region analysis to take into account backward execution.
  - Extend the supported subset of Mercury.