This is a two-in-one summary.

The first paper, *BI as an Assertion Language* [1], (apparently) introduces the magic wand operator $-\!\!*$ and gives a semantics for freeing memory. Magic wands are implications with a fresh piece of memory as their premise.[1] The semantics for freeing memory came as a consequence of giving a classical model for Reynolds' then-unnamed intuitionistic logic about heaps & pointers.

The second paper, Reynolds's survey on separation logic [2], distills ~ 2 years of research into a smooth introduction with many examples. It contains no theorems or "new" results, but rather demonstrates separation logic specifications & simple proofs. Also, the paper coins the term "separation logic", emphasizing that splitting the heap (or generally, resource) into disjoint sections is the key insight.

Strengths

- Separation logic is very important. If I have to talk about memory, this is the specification langauge I want to use. And these papers started it all. (I hope today's presentation gives an update on how much of Reynolds 3-page future work section has been checked off, and what new ideas have come up along the way.)
- The BI paper is going over my head now, but the LICS paper is very easy to understand.

Weaknesses

- Equating datatypes (lists & trees) with propositions about the heap layout upset me. Different types can have similar in-memory representations! Equating these types in separation logic will break my type safety guarantees.

  My takeway from those sections is that I hope separation logic become obselete. It's very good that we have it today, to validate the many C programs that are born and exist, but in the future I hope we rely on semantics-preserving compilers and do our formal methods in a higher-level language.
- Can we use separation logic to talk about the physical layout of machines? For instance, reasoning about address spaces and pages swapped out to disk—giving $*$ a physical semantics. This seems like it would be an easy connection to make after extolling modular reasoning and I'm surprised neither paper mentions it.
- Now that I think of locality and caches, I remember that coherence is an important issue. Separation logic does not allow disjoint & overlapping memory, but this would be important to model in multicore or distributed settings—"coherent" overlaps with a specification for resolving conflicts.

  This also sounds very hard. We suddenly have two unique stores (references in two caches) that aren't exactly shared, but happen to correspond to one store in the main memory.

I can't resist making two typesetting comments—one weakness, one strength:

- What happened to the margins in the BI paper?
- John Reynolds was very good at making diagrams.

References

[1] Samin Ishtiaq and Peter O'Hearn. BI as an assertion language for mutable data structures. In *POPL*, 2001.

[2] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, 2002.

[1] The authors of both papers especially note that $A -\!\!* B$ does not imply $A \Rightarrow B$ i.e. this is a novel operator and not just a specialization of linear logic's $-\!\!\circ$.