Fragment Abstraction: Singly Linked Lists sec:fragment-abstraction In this section, we describe in more detail our fragment abstraction for concurrent programs that operate on a shared heap. We consider a program with global variables and thread-local variables . We assume that all global variables are pointer variables. We describe our fragment abstractions for three classes of heap structures: in the following subsection, we consider programs operating on singly-linked lists, in Subsection subsect:skiplists, we consider programs operating on skiplists, and in Subsection subsect:skiplists, we consider programs operating arrays of singly linked lists.

Fragment Abstraction for Singly-Linked List-Based Programs subsec:frag-sll In this subsection, we describe our symbolic representation, using fragment abstraction, for programs that operate on singly-linked lists (SLLs). This representation is also the basis for our representation for programs operating on skiplists, described in Subsection subsect:skiplists and programs operating on arrays of SLLs, in Subsection subsect:skiplists.

Say that for now, we ignore timers?

We assume that we must analyze the product of a program and an observer. The program operates on a heap, where each cell has exactly one pointer field, named `next`, and at most one `data` field, which assumes values from the same domain as observer registers.

Introduce notation for the set of local -variables?. Maybe defined which threads we talk about? For now, we skip timestamps.

We first define our data abstraction. For each thread-local variable, and each non-pointer cell field, which ranges over some concrete domain, we define a corresponding abstract domain, as follows. itemize

- F or small concrete domains (including that of the program counter), the abstract domain is the same as the concrete one.
- F or locks variables and lock fields, the abstract domain is $me, other, free$. Should we explain?
- F or the concrete domain of data values, the abstract domain is the set of mappings from local variables ranging over and observer registers to the set $<, =, >$. An element in the abstract domain represents a concrete data value `d` if it maps each local variable and observer register with value $\mathtt{d}'$ to a set which includes a relation $\sim$ such that $\mathtt{d} \sim \mathtt{d}'$.