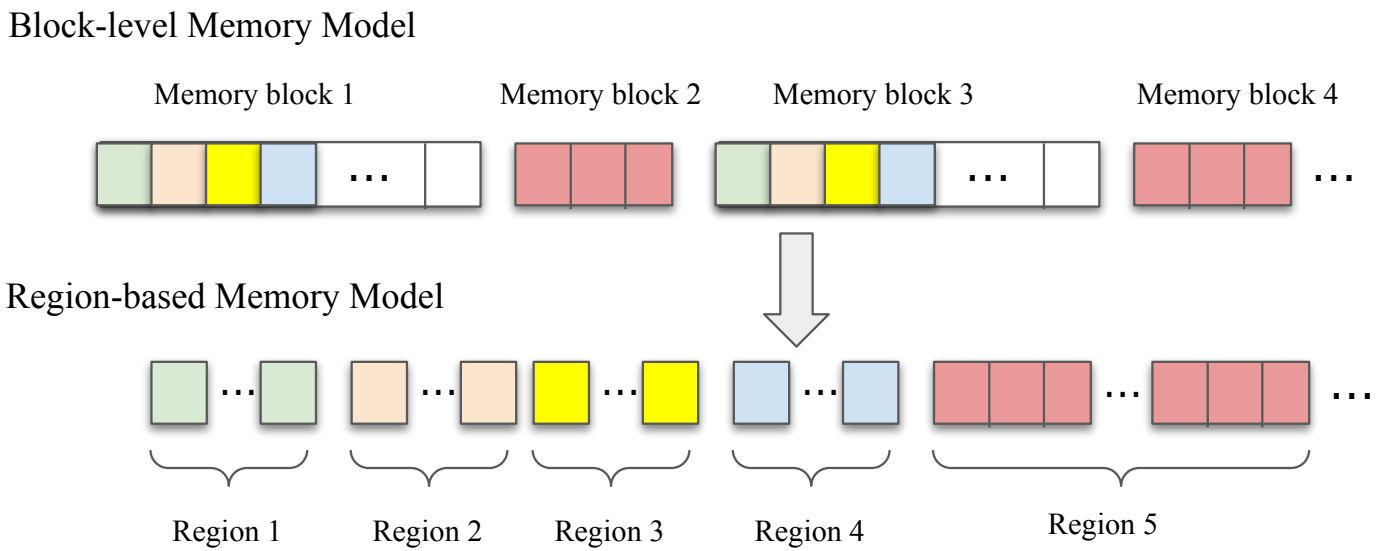
CrabIR

- A goto-based control flow language via `assume` and `goto` statements and without ϕ nodes
- Strongly-typed
- Standard arithmetic/Boolean/comparison operations over integers and booleans:
  - Explicit casts between Boolean and integers
- Function without side-effects
- Pointers represented by a region-based memory model
  - Explicit casts between integers and references
- Unbounded arrays whose addresses cannot be taken: disjoint from each other and other regions

CrabIR Region-Based Memory Model

- Word-addressable: all data location has the same size and all pointer addresses are divisible by the word size



- A block slice is a subsequence of bytes within a memory block
- A region consists of sliced memory blocks
- A reference is an address within a region

CrabIR Syntax

- Integer variable $v_i \in \mathcal{V}_{\mathcal{I}}$
- Boolean variable $v_b \in \mathcal{V}_{\mathcal{B}}$
- Reference variable $v_{ref} \in \mathcal{V}_{\mathcal{R}ef}$
- Region variable $v_{rgn} \in \mathcal{V}_{\mathcal{R}gn}$
- Array variable $v_A \in \mathcal{V}_{\mathcal{A}}$
- $\mathcal{V}_{\mathcal{B}} \cap \mathcal{V}_{\mathcal{I}} \cap \mathcal{V}_{\mathcal{R}ef} \cap \mathcal{V}_{\mathcal{R}gn} \cap \mathcal{V}_{\mathcal{A}} = \emptyset$
- v_n denotes a Boolean or integer variable v_i or v_b
- v_s denotes a scalar variable v_i, v_b, or v_{ref}

> We assume that each variable is explictly typed, and therefore, we omit types from statements

CrabIR Syntax: Goto-based Control Flow

$$
\begin{array}{lcl}
P & \rightarrow & F^* \\
F & \rightarrow & \texttt{declare}\ \mathit{fun}(v^*)\ B^+ \\
B & ::= & \text{label} : \mathit{Stmt}^*\ T \\
T & ::= & \texttt{goto}\ \text{label}^+ \mid \texttt{return}\ v^* \\
\mathit{Stmt} & ::= & \mathit{Stmt}_{int} \mid \mathit{Stmt}_{bool} \mid \mathit{Stmt}_{rgn} \mid \mathit{Stmt}_{array} \mid \\
 & & [\ v^+ :=]\ \texttt{call}\ \mathit{fun}(v^+) \mid \\
 & & v := \texttt{havoc}\ () \mid \\
 & & \texttt{assume}\ (v_b) \mid \\
 & & \texttt{assert}\ (v_b)
\end{array}
$$

CrabIR Syntax: Arithmetic Operations

$$
\begin{array}{lll}
Stmt_{arith} & ::= & v_i := a \\
 & & v_i := \texttt{sext}\ (v_i)\ \mid \\
 & & v_i := \texttt{zext}\ (v_i)\ \mid \\
 & & v_i := \texttt{trunc}\ (v_i)\ \mid \\
 & & v_b := \texttt{inttobool}\ (v_i) \\
 & & v_i := \texttt{booltoint}\ (v_b) \\
a & ::= & n\ \mid v_i\ \mid a_1\ op_a\ a_n
\end{array}
$$

Note that we omit bitwidth because it is deduced from v_i (arguments and returns)

CrabIR Syntax: Boolean Operations

$$
\begin{array}{lll}
Stmt_{bool} & ::= & v_b := b \\
b & ::= & \text{true} \mid \text{false} \mid \\
 & & \neg\ b \mid b_1\ op_b\ b_2 \mid \\
 & & a_1\ op_r\ a_2 \mid b_{ref}
\end{array}
$$

CrabIR Syntax: Arrays

$$
\begin{array}{lll}
Stmt_{array} & ::= & v_n := \texttt{arrayread}\ (v_A, v_i) \ \mid \\
 & & \texttt{arraywrite}\ (v_A, v_i, v_n) \ \mid \\
 & & v_A := \texttt{arrayassign}(v_A)
\end{array}
$$

Note that only Boolean or integer variables are stored in arrays

CrabIR Syntax: Regions and References

$$
\begin{array}{lll}
Stmt_{rgn} & ::= & \texttt{initrgn}\ (v_{rgn}) \ \mid \\
 & & v_{rgn} := \texttt{copyrgn}(v_{rgn}) \ \mid \\
 & & v_{ref} := \texttt{makeref}\ (v_{rgn},\ v_i) \ \mid \\
 & & \texttt{freeref}\ (v_{rgn},\ v_{ref}) \ \mid \\
 & & v_s := \texttt{loadref}\ (v_{rgn},\ v_{ref}) \ \mid \\
 & & \texttt{storeref}\ (v_{rgn},\ v_{ref},\ v_s) \ \mid \\
 & & (v_{rgn}, v_{ref}) := \texttt{gepref}\ (v_{rgn},\ v_{ref}, v_i) \\
 & & v_i := \texttt{reftoint}\ (v_{rgn},\ v_{ref}) \ \mid \\
 & & v_{ref} := \texttt{inttoref}\ (v_{rgn},\ v_i) \ \mid \\
b_{ref} & := & v_{ref} = v_{ref} \ \mid \texttt{isnull}(v_{ref})
\end{array}
$$