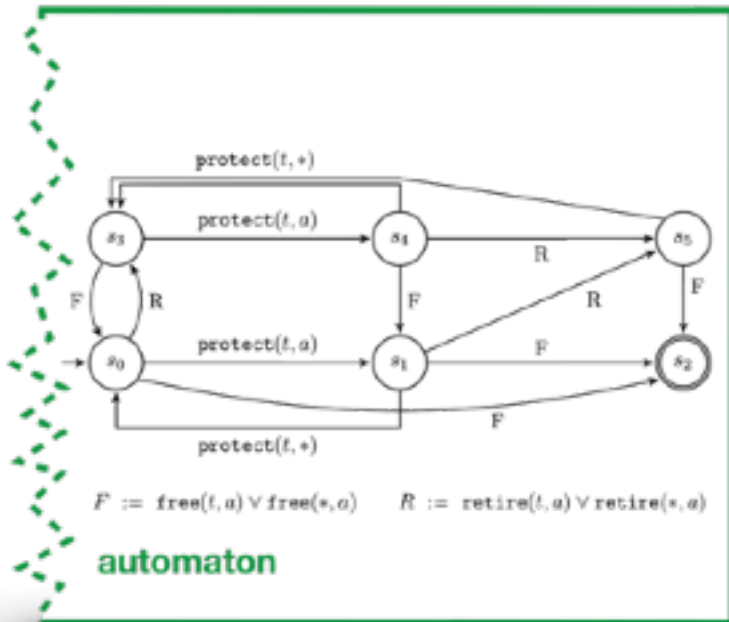


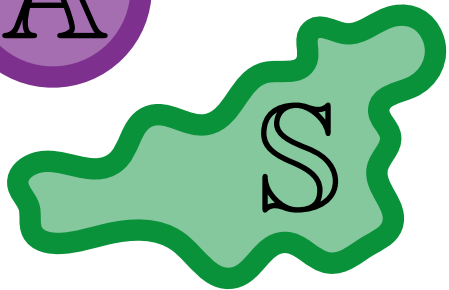
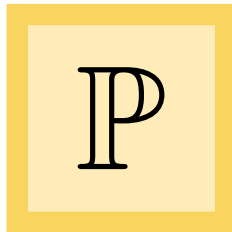


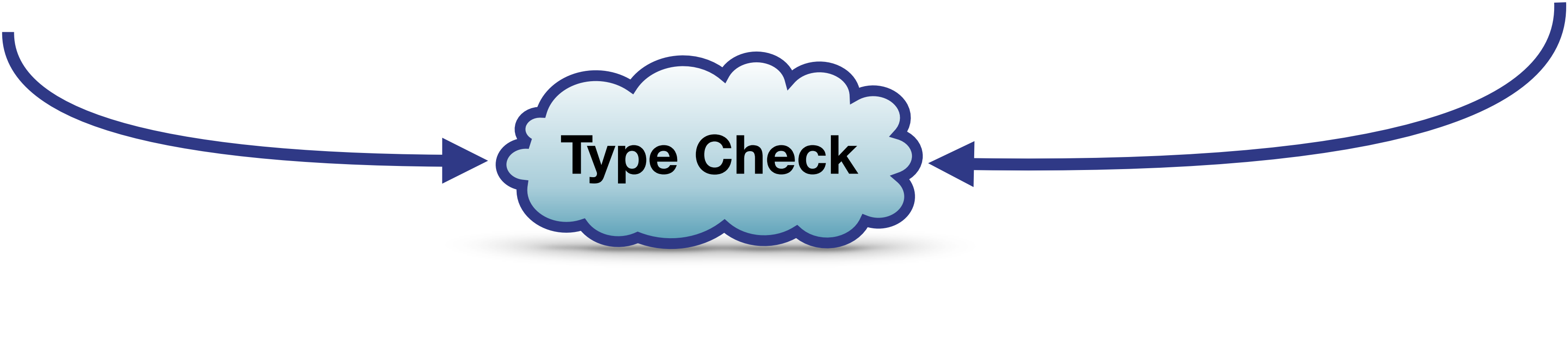
off-the-shelf verifier

Final Approach

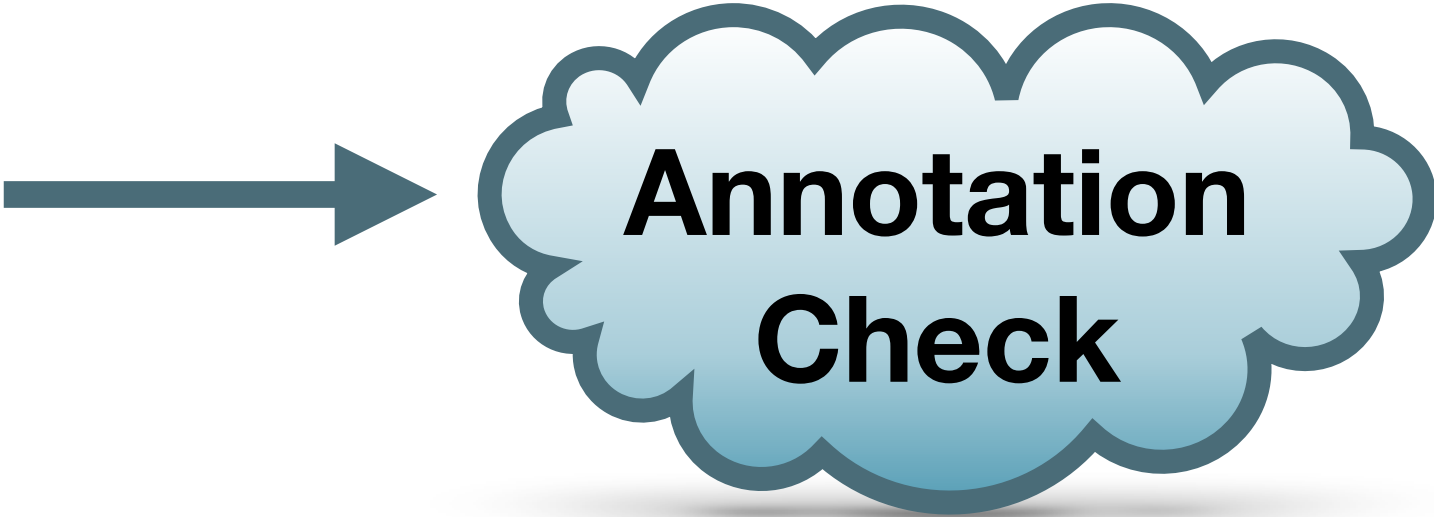


Types:

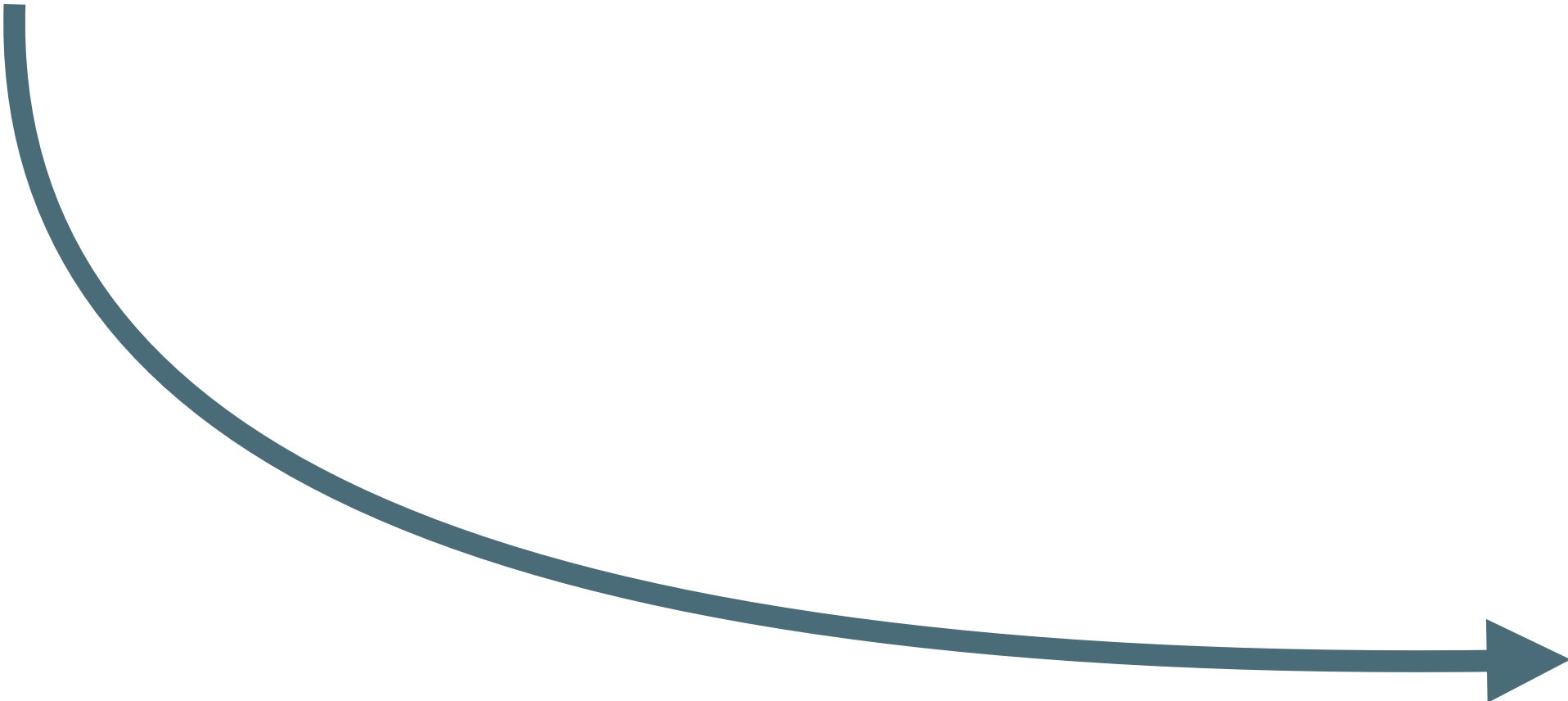








**Annotation
Check**



```
struct Node {
    data_t data;
    Node* next;
}

shared:
Node* Head;
Node* Tail;

void init() {
    Head = new Node();
    Head->next = null;
    Tail = Head;
}

void enqueue(data_t val) {
    Node* node = new Node();
    node->data = val;
    node->next = null;
    while (true) {
        Node* tail = Tail;

        Node* next = tail->next;
        if (Tail != tail) continue;
        if (next == null) {
            if (CAS(&tail->next, null, node)) {
                CAS(&Tail, tail, node);
            } else {
                CAS(&Tail, tail, next);
            }
        }
    }
}

data_t dequeue() {
    while (true) {
        Node* head = Head;
        Node* tail = Tail;
        Node* next = head->next;

        if (head != head) continue;
        if (head == tail) {
            if (next == null) return empty_T;
            else CAS(&Tail, tail, next);
        } else {
            data = head->data;
            if (CAS(&Head, head, next)) {
                return data;
            }
        }
    }
}
```

queue
lock-free

GC

instrumentation

```

struct Node {
    data_t data;
    Node* next;
    Node* next;
}

shared:
Node* head;
Node* tail;

void init() {
    head = new Node();
    head->next = null;
    tail = head;
}

data_t dequeue() {
    while (true) {
        Node* head = head;
        protect(head);
        if (head != head) continue;
        Node* tail = tail;
        Node* next = head->next;
        protect(next);
        if (head != head) continue;
        if (tail != tail) continue;
        if (next == null) {
            if (CAS(head->next, null, null)) {
                CAS(tail, tail, next);
            }
        } else {
            CAS(tail, tail, next);
        }
    }
}

}

```

queue
lock-free **SMR**

```

struct Bar {
    Bar* next;
    Node* hp0;
    Node* hp1;
}

shared:
Bar* HPHead;

thread-local:
Bar* myHps;
List*Nodes = retiredList;

void join() {
    myHps = new HPHead();
    while (true) {
        Bar* tmp = HPHead;
        myHps->next = tmp;
        if (CAS(HPHead, tmp, myHps)) {
            break;
        }
    }
}

void part() {
    unprotect();
    unprotect();
}

```

lock-free **HP**
no reclamation

```

void protect(Node* ptr) {
    myHps->hp0 = ptr;
}

void protect(Node* ptr) {
    myHps->hp1 = ptr;
}

void retireNode(Node* ptr) {
    retiredList.add(ptr);
    if (!reclaim())
        return;
}

void reclaim() {
    List*Nodes = protectedList;
    Bar* tmp = HPHead;
    while (tmp != null) {
        Node* hp0 = cur->hp0;
        Node* hp1 = cur->hp1;
        protectedList.add(hp0);
        protectedList.add(hp1);
        cur = cur->next;
    }
    for (Node* ptr : retiredList) {
        if (!protectedList.contains(ptr)) {
            retiredList.remove(ptr);
            delete ptr;
        }
    }
}

```



```
struct Node {          shared:      void init() {
    data_t data;         Node* Head;      Head = new Node();
    Node* next;         Node* Tail;     Head->next = null;
}                          Tail = Head;
                          }

void enqueue(data_t val) {      data_t dequeue() {
    Node* node = new Node();    while (true) {
    node->data = val;           Node* head = Head;
    node->next = null;
    while (true) {
        Node* tail = Tail;

        Node* next = tail->next;
        if (Tail != tail) continue;
        if (next == null) {
            if (CAS(tail->next, null, node)) {
                CAS(Tail, tail, node);
            }
        } else {
            CAS(Tail, tail, next);
        }
    }
}

void init() {
    Head = new Node();
    Head->next = null;
    Tail = Head;
}

data_t dequeue() {
    while (true) {
        Node* head = Head;
        Node* next = head->next;
        if (head != head) continue;
        if (head == tail) {
            if (next == null) return empty_t;
            else CAS(Tail, tail, next);
        } else {
            data = head->data;
            if (CAS(Head, head, next)) {
                return data;
            }
        }
    }
}
```

queue
lock-free

GC



Thanks





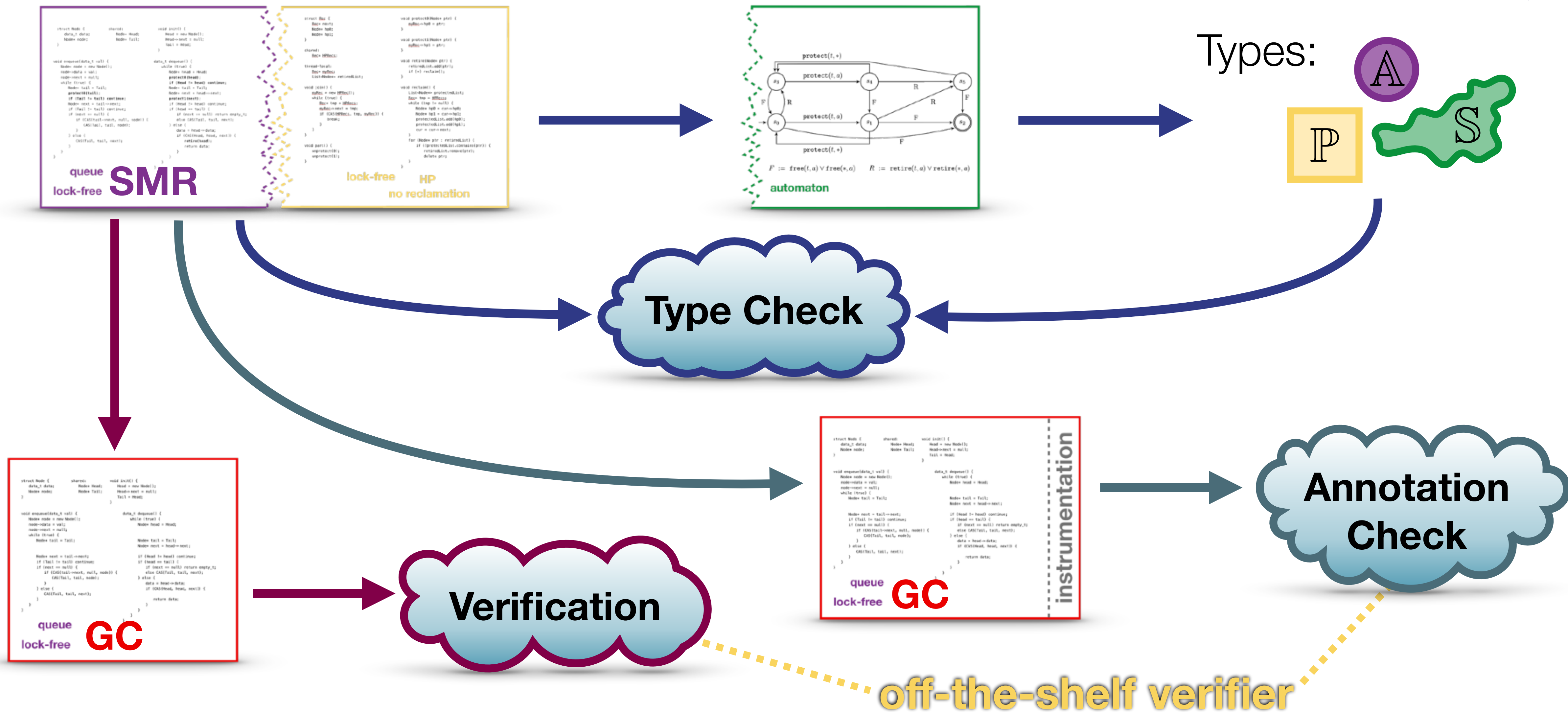






Thanks

Final Approach



Fin.