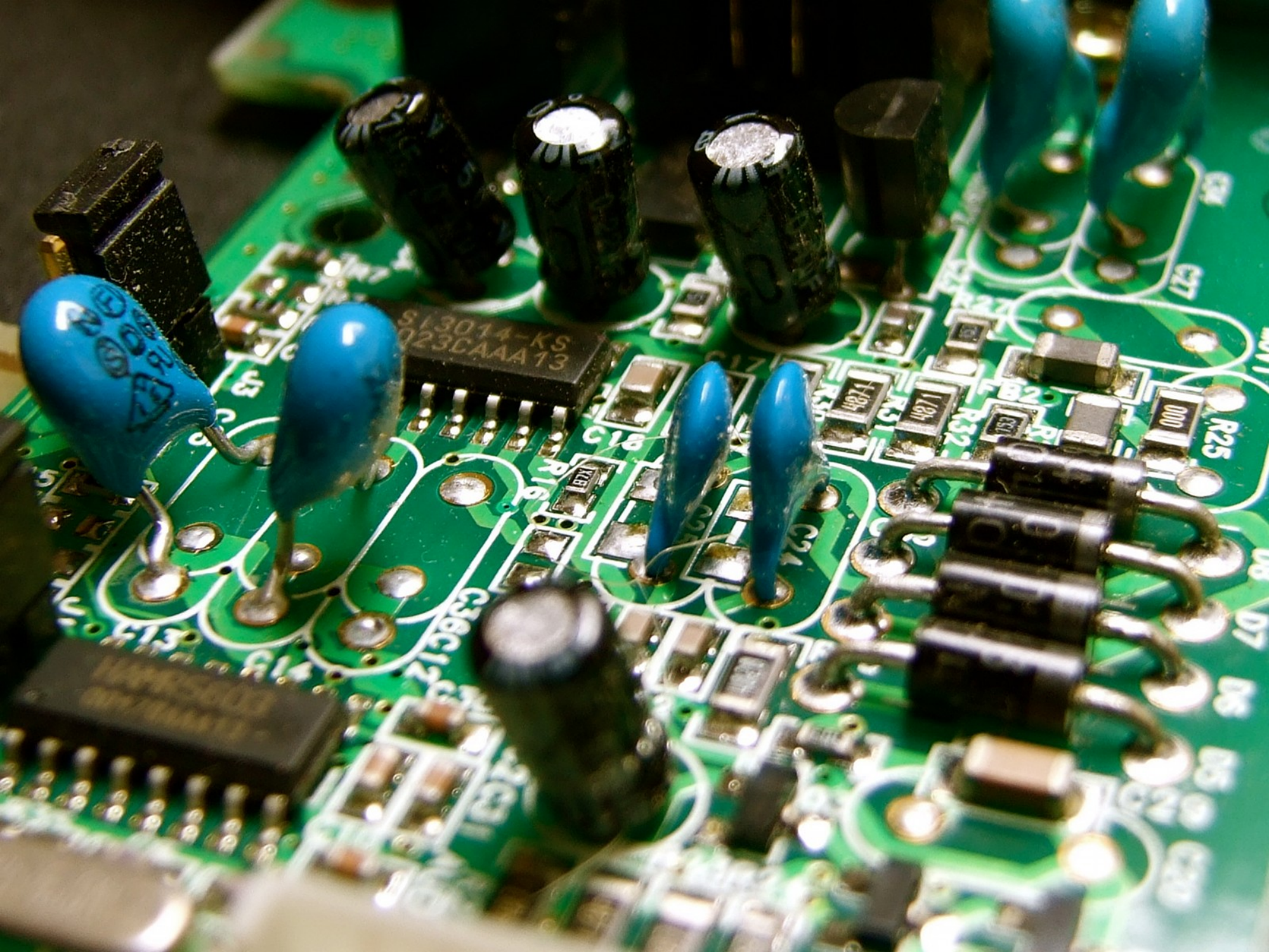
Si3014-KS
923CAAA13
R25

Notace

Už známe

a . . . adresa nějakého objektu

$h(a)$ . . . objekt na adrese a v haldě h

$field(h(a), f)$ . . . hodnota atributu f objektu na adrese a v haldě h

Zjednodušení notace

$h(a).f \equiv field(h(a), f)$ . . . hodnota atributu f objektu na adrese a v haldě h

$h(a.f) \equiv h(field(h(a), f))$ . . . objekt na adrese uložené v atributu f objektu na adrese a v haldě h

$h(a.f1).f2 \equiv field(h(field(h(a), f1)), f2)$ . . . hodnota atributu $f2$ objektu na adrese uložené v atributu $f1$ objektu na adrese a v haldě h

$h(a.f1.f2) \equiv h(field(h(field(h(a), f1)), f2))$ . . . objekt na adrese uložené v atributu $f2$ objektu na adrese uložené v atributu $f1$ objektu na adrese a v haldě h

Správnost kódu

Kód

```
class Node {
    int val;
    Node next;
    boolean contains(int v) {
        if (val == v) return true;
        else return next == null ? false : next.contains(v);
    }
}

class LinkedList {
    Node first;
    boolean isEmpty() { return first == null; }
    boolean contains(int v) {
        return first == null ? false : first.contains(v);
    }
}
```

Metoda `contains` pracuje správně

$$elems : addr \times heap \to num^*$$

$$elems(n, h) = \begin{cases} \langle\rangle \text{ pokud } n = null \\ \langle v \rangle \cdot elems(a, h) \text{ pokud } h(n) = Node(val = v, next = a) \end{cases}$$

$$(h, s, g) \vdash a.contains(v) \mapsto^* true$$

$$\Updownarrow$$

$$elems(h(a).first, h) = \langle z_1, \ldots, z_k \rangle \wedge v \in \{z_1, \ldots, z_k\}$$

Metoda `contains` pracuje správně (II)

$$(h, s, g) \vdash a.contains(v) \mapsto^* true$$

$$\Updownarrow$$

$$elems(h(a).first, h) = \langle z_1, \ldots, z_k \rangle \wedge v \in \{z_1, \ldots, z_k\}$$

$a.contains(v)$ volané nad haldou h vrátí *true*

tehdy a právě tehdy, pokud

sekvence hodnot v seznamu odkazovaném proměnnou a obsahuje hodnotu v

Několik aspektů důkazu správnosti

- Metoda vrátí správnou hodnotu.
- Metoda nespadne (dereferencování *null*, indexování mimo meze polí, špatné přetypování, dělení nulou, ..., tzn. podtypy `RuntimeException`).
- Metoda doběhne v konečném čase.

Dokazovat tato podtvrzení odděleně je většinou jednodušší než dokazovat je dohromady.

Aserce

```
class Node {
    int val; Node next;

    void remove(int v) {
        if (next == null) return;
        if (next.val == v)
            next = next.next;
        next.remove(v);

    }

    /* dalsi metody */
}
```

```
class LinkedList {
    Node first;

    void remove(int v) {
        if (first == null) return;
        if (first.val == v)
            first = first.next;
        first.remove(v);
        assert !contains(v);
    }

    /* dalsi metody */
}
```

Aserce

```
class Node {
    int val; Node next;

    void remove(int v) {
        if (next == null) return;
        if (next.val == v) {
            next = next.next;
            remove(v);
        }
        else next.remove(v);

    }

    /* dalsi metody */
}
```

```
class LinkedList {
    Node first;

    void remove(int v) {
        if (first == null) return;
        if (first.val == v) {
            first = first.next;
            remove(v);
        }
        else first.remove(v);
        assert !contains(v);
    }

    /* dalsi metody */
}
```

Invariant objektu

```
class Node {
    int val;
    Node next;
    boolean contains(int v) {
        if (val == v) return true;
        if (val > v) return false;
        return next == null ? false : next.contains(v);
    }
}

class LinkedList {
    Node first;
    boolean contains(int v) {
        return first == null ? false : first.contains(v);
    }
}
```

Zpřístupnění vnitřní reprezentace objektu

```
class Node {
    int val;
    Node next;
    /* metody */
}

class LinkedList {
    Node first;
    Node getNodes() { return first; }
    void add(Node n) {
        n.next = first;
        first = n;
    }
    /* metody */
}
```

Zapouzdření

- Uživatel se nemusí starat o vnitřní implementaci objektu.
- Uživatel nesmí záviset na vnitřní implementaci objektu.
- Uživatel nesmí mít možnost rozbít vnitřní implementaci objektu.