Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen
Sebastian Ullrich sebastian.ullrich@kit.edu
Max Wagner maximilian.wagner@kit.edu

Semantik von Programmiersprachen – SS 2019

http://pp.ipd.kit.edu/lehre/SS2019/semantik

Lösungen zu Blatt 13: Axiomatische Semantik Besprechung: 22.07.2019

1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

(a) $\{\,\texttt{x == 0}\,\}\,\texttt{x := x + 1}\,\{\,\texttt{0 <= x}\,\}$ ist eine Zusicherung.

(b) In $\{\,\texttt{x} = n\,\}\,\texttt{y := x * n}\,\{\,\texttt{y} = n^2\,\}$ ist n eine logische Variable.

(c) Wenn $\vdash \{\,P\,\}\,\texttt{skip}\,\{\,Q\,\}$, dann $P = Q$.

(d) Es gibt ein c mit $\vdash \{\,\mathbf{tt}\,\}\,c\,\{\,\mathbf{ff}\,\}$.

(e) Statt der Regel Ass_P wäre auch die Regel $\vdash \{\,P\,\}\,x\ \texttt{:=}\ a\,\{\,P[x \mapsto \mathcal{A}\,[\![a]\!]]\,\}$ sinnvoll.

Lösung:

(1a) Falsch. Eine Zusicherung besteht aus Vorbedingung, Anweisung und Nachbedingung, wobei die Bedingungen *Zustandsprädikate* sind, also Funktionen des Typs $\Sigma \to \mathbb{B}$. Unsere Notationsregeln erlauben, λ-Abstraktionen und Zustandsanwendungen auf Variablen wegzulassen. `x == 0` und `0 <= x` sind aber boolesche Ausdrücke Bexp. Eine Zusicherung wäre z.B. $\{\,\texttt{x} = 0\,\}\,\texttt{x := x + 1}\,\{\,0 \leq \texttt{x}\,\}$

(1b) Falsch. Logische Variablen dürfen im Programm nicht vorkommen.

(1c) Falsch. Bei der Regelinversion auf $\vdash \{\,\cdot\,\}\,\texttt{skip}\,\{\,\cdot\,\}$ gibt es zwei Fälle: Skip_P und Cons_P! Damit gilt dann z.B.:

$$\dfrac{\mathbf{ff} \implies \mathbf{tt} \qquad \dfrac{}{\vdash \{\,\mathbf{tt}\,\}\,\texttt{skip}\,\{\,\mathbf{tt}\,\}}\ \text{Skip}_\text{P} \qquad \mathbf{tt} \implies \mathbf{tt}}{\vdash \{\,\mathbf{ff}\,\}\,\texttt{skip}\,\{\,\mathbf{tt}\,\}}\ \text{Cons}_\text{P}$$

(1d) Richtig. Zum Beispiel $c = \texttt{while (true) do skip}$:

$$\dfrac{\dfrac{}{\vdash \{\,\mathbf{tt}\,\}\,\texttt{skip}\,\{\,\mathbf{tt}\,\}}\ \text{Skip}_\text{P}}{\vdash \{\,\mathbf{tt}\,\}\,c\,\{\,\mathbf{ff}\,\}}\ \text{While}_\text{P}$$

(1e) Falsch. Denn damit ließen sich offensichtlich unsinnige Zusicherungen ableiten, zum Beispiel $\vdash \{\,\texttt{x} = 3\,\}\,\texttt{x := 5}\,\{\,\mathbf{ff}\,\}$, denn

$$(\lambda\sigma.\ \sigma(\texttt{x}) = 3)[\texttt{x} \mapsto \mathcal{A}\,[\![5]\!]] = (\lambda\sigma.\ (\sigma[\texttt{x} \mapsto 5])(\texttt{x}) = 3) = \lambda\sigma.\ 5 = 3 = \lambda\sigma.\ \mathbf{ff}$$

2. Quadratwurzelberechnung (H)

Zeigen Sie, dass folgende Zusicherung in der axiomatischen Semantik ableitbar ist.

$\{\,\texttt{a} \geq 0\,\}$
```
n := 0; m := 1; k := 0;
while (k <= a - 1) do (n := n + 1; k := k + m; m := m + 2)
```
$\{\,\texttt{n} - 1 < \sqrt{\texttt{a}} \leq \texttt{n}\,\}$

Hinweis: Die Invariante der Schleife ist etwas trickreich. Berechnen Sie ein paar einfache Beispiele und versuchen Sie herauszufinden, in welcher Beziehung `m`, `n` und `k` zueinander stehen.

Lösung: Die Implementierung nutzt aus, dass die Differenzen von je zwei aufeinanderfolgenden Quadratzahlen die Folge der ungerade Zahlen bilden:

$$1^2 - 0^2 = 1$$
$$2^2 - 1^2 = 3$$
$$3^2 - 2^2 = 5$$
$$4^2 - 3^2 = 7$$

Dies folgt übrigens aus der binomischen Formel $(n+1)^2 - n^2 = 2n + 1$. Entsprechend ist n^2 die Summe der ersten n ungeraden Zahlen.

In der Schleife durchläuft `m` diese Folge der ungeraden Zahlen. `k` summiert diese auf, ist also immer die entsprechende Quadratzahl. Damit ergibt sich die Invariante

$$\mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}}$$

mit der folgende Ableitung gelingt:

$\{ \mathtt{a} \geq 0 \}$
$\Longrightarrow$ $\textsc{Cons}_\textsc{P}$
$\{ 0 = 0^2 \wedge 1 = 2 \cdot 0 + 1 \wedge 0 - 1 < \sqrt{\mathtt{a}} \}$
`n := 0` $\textsc{Ass}_\textsc{P}$
$\{ 0 = \mathtt{n}^2 \wedge 1 = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$
`m := 1` $\textsc{Ass}_\textsc{P}$
$\{ 0 = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$
`k := 0` $\textsc{Ass}_\textsc{P}$
$\{ \mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$
`while (k <= a - 1) do (n := n + 1; k := k + m; m := m + 2)` A (s.u.) mit $\textsc{While}_\textsc{P}$
$\{ \mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \wedge \mathtt{k} > \mathtt{a} - 1 \}$
$\Longrightarrow$ $\textsc{Cons}_\textsc{P}$
$\{ \mathtt{n} - 1 < \sqrt{\mathtt{a}} \leq \mathtt{n} \}$

A:
$\{ \mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \wedge \mathtt{k} \leq \mathtt{a} - 1 \}$
$\Longrightarrow$ $\textsc{Cons}_\textsc{P}$
$\{ \mathtt{k} + \mathtt{m} = (\mathtt{n}+1)^2 \wedge \mathtt{m} + 2 = 2 \cdot (\mathtt{n}+1) + 1 \wedge (\mathtt{n}+1) - 1 < \sqrt{\mathtt{a}} \}$
`n := n + 1` $\textsc{Ass}_\textsc{P}$
$\{ \mathtt{k} + \mathtt{m} = \mathtt{n}^2 \wedge \mathtt{m} + 2 = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$
`k := k + m` $\textsc{Ass}_\textsc{P}$
$\{ \mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} + 2 = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$
`m := m + 2` $\textsc{Ass}_\textsc{P}$
$\{ \mathtt{k} = \mathtt{n}^2 \wedge \mathtt{m} = 2 \cdot \mathtt{n} + 1 \wedge \mathtt{n} - 1 < \sqrt{\mathtt{a}} \}$

3. Schnelle Division (Ü)

Arithmetische Ausdrücke in Aexp enthalten keinen Divisionsoperator. In dieser Aufgabe sollen Sie ein While-Programm schreiben, das zwei ganze Zahlen mit logarithmischer Laufzeit (in der Größe der Zahlen) dividiert, und dieses korrekt beweisen.

(a) Ergänzen Sie folgendes Programmskelett, sodass das vollständige Programm c folgende Spezifikation in Form einer Zusicherung erfüllt, die angegebenen Invarianten verwendet und die Eingabevariablen `x` und `y` unverändert lässt:

$$\{ \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \}\, c \,\{ \mathtt{x} = \mathtt{q} \cdot \mathtt{y} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{y} \}$$

`... ;` // Invariante I_0 sicherstellen
`while (z <= x) do ( ...` // Invariante I_0 erhalten und `z` verdoppeln`);`
`... ;` // Invariante I_1 sicherstellen
`while (not (n == 0)) do ( ...` // Invariante I_1 erhalten und `n` verkleinern`)`

wobei

$$I_0 = (\mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}}) \wedge \mathtt{n} \geq 0 \wedge \mathtt{x} \geq 0 \wedge \mathtt{y} > 0$$
$$I_1 = (\mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r}) \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge (\mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}}) \wedge \mathtt{n} \geq 0$$

Für diese Aufgabe können Sie neben den bisherigen arithmetischen Ausdrücken auch den neuen Verschiebe-Ausdruck a `» 1` mit der Semantik

$$\mathcal{A}[\![a \texttt{ » } 1]\!]\sigma = \left\lfloor \frac{\mathcal{A}[\![a]\!]\sigma}{2} \right\rfloor$$

verwenden, wobei $\lfloor x \rfloor$ die rationale Zahl x auf die nächstkleinere ganze Zahl abrundet.

(b) Weisen Sie nach, dass obige Zusicherung für Ihr Programm c ableitbar ist. Verwenden Sie dabei die angegebenen Invarianten für die Schleifen.

(c) Terminiert Ihr Programm immer?

(d) Eliminieren Sie jetzt den Operator a `» 1` in Ihrem Programm, verwenden Sie bei Bedarf Schleifen. Wie müsste man den Korrektheitsbeweis anpassen? In welcher Komplexitätsklasse liegt Ihr Programm jetzt?

Lösung:

(3a) Man kann schrittweise aus den Invarianten die Aktionen der Schleife konstruieren. Das vollständige Programm ist dann:

```
n := 0; z := y
while (z <= x) do (z := 2 * z; n := n + 1);
q := 0; r := x;
while (not (n == 0)) do (
    n := n - 1;
    q := q * 2;
    z := z » 1;
    if (z <= r) then q := q + 1; r := r - z else skip
)
```

(3b)

$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \,\}$
`n := 0` $\quad$ ASS$_P$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} = 0 \,\}$
`z := y` $\quad$ ASS$_P$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} = 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \,\}$
$\Longrightarrow$ $\quad$ CONS$_P$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \,\}$
`while (z <= x) do (z := 2 * z; n := n + 1)` $\quad$ A (s.u.) mit Regel WHILE$_P$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{z} > \mathtt{x} \,\}$
`q := 0` $\quad$ ASS$_P$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{z} > \mathtt{x} \wedge \mathtt{q} = 0 \,\}$
`r := x` $\quad$ ASS$_P$

$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{z} > \mathtt{x} \wedge \mathtt{q} = 0 \wedge \mathtt{r} = \mathtt{x} \,\}$
$\Longrightarrow$ $\textsc{Cons}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \,\}$
`while (not (n == 0)) do` c' B (s.u.) mit Regel $\textsc{While}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} = 0 \,\}$
$\Longrightarrow$ $\textsc{Cons}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{y} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{y} \,\}$

A:
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{z} \leq \mathtt{x} \,\}$
$\Longrightarrow$ $\textsc{Cons}_{\mathrm{P}}$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \,\}$
`z := 2 * z` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}+1} \,\}$
`n := n + 1` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 1 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \,\}$
$\Longrightarrow$ $\textsc{Cons}_{\mathrm{P}}$
$\{\, \mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \,\}$

B:
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \wedge \mathtt{n} \neq 0 \,\}$
`n := n - 1` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}+1} \wedge \mathtt{n} \geq 0 \,\}$
`q := q * 2` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} = \frac{\mathtt{q}}{2} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}+1} \wedge \mathtt{n} \geq 0 \,\}$
`z := z » 1` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \,\}$
`if (z <= r) then q := q + 1; r := r - z else skip` C und D (s.u.) mit $\textsc{If}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \,\}$

C:
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} \leq \mathtt{r} \,\}$
`q := q + 1` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + (\mathtt{r} - \mathtt{z}) \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} \leq \mathtt{r} \,\}$
`r := r - z` $\textsc{Ass}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge -\mathtt{z} \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \wedge 0 \leq \mathtt{r} \,\}$
$\Longrightarrow$ $\textsc{Cons}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \,\}$

D:
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \wedge \mathtt{z} > \mathtt{r} \,\}$
`skip` $\textsc{Skip}_{\mathrm{P}}$
$\{\, \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \,\}$

(3c) Ja, aber mit partieller Korrektheit lässt sich dies nicht beweisen. Dafür bräuchte man ein Kalkül für totale Korrektheit. Dies würde für jede Schleife eine Abstiegsfunktion fordern, die während jeder Schleifeniteration strikt kleiner wird. Bei den beiden Schleifen nähme man z.B. $\mathtt{x} - \mathtt{n}$ bzw. $\mathtt{n}$.

(3d) `z » 1` lässt sich nur mit einer Schleife eliminieren. Statt dies naiv mit Subtraktion zu lösen, kann man hier aber ausnutzen, dass $\mathtt{n}$ den Logarithmus zur Basis 2 von $\frac{\mathtt{z}}{\mathtt{y}}$ enthält:

```
n := 0; z := y
while (z <= x) do (z := 2 * z; n := n + 1);
q := 0; r := x;
while (not (n == 0)) do (
    n := n - 1;
    q := q * 2;
    u := y; i := 0; while (not (i == n)) do (i := i + 1; u := 2 * u); z := u
    if (z <= r) then q := q + 1; r := r - z else skip
)
```

Den Korrektheitsbeweis muss man nur an der geänderten Stelle ändern. Statt der Regel Ass_P für `z := z » 1` muss nun folgendes Stück stehen, wobei

$$P = \mathtt{x} = \frac{\mathtt{q}}{2} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}+1} \wedge \mathtt{n} \geq 0$$

$\{ P \}$
`u := y` Ass_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \}$
`i := 0` Ass_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \wedge \mathtt{i} = 0 \}$
$\Longrightarrow$ Cons_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}} \wedge \mathtt{i} \geq 0 \}$
`while (not (i == n)) do (i := i + 1; u := 2 * u)` E (s.u) mit While_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}} \wedge \mathtt{i} \geq 0 \wedge \mathtt{i} = \mathtt{n} \}$
$\Longrightarrow$ Cons_P
$\{ \mathtt{x} = \mathtt{q} \cdot \mathtt{u} + \mathtt{r} \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{u} \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \}$
`z := u` Ass_P
$\{ \mathtt{x} = \mathtt{q} \cdot \mathtt{z} + \mathtt{r} \wedge 0 \leq \mathtt{r} < 2 \cdot \mathtt{z} \wedge \mathtt{z} = \mathtt{y} \cdot 2^{\mathtt{n}} \wedge \mathtt{n} \geq 0 \}$

E:
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}} \wedge \mathtt{i} \geq 0 \wedge \mathtt{i} \neq \mathtt{n} \}$
`i := i + 1` Ass_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}-1} \wedge \mathtt{i} \geq 1 \wedge \mathtt{i} - 1 \neq \mathtt{n} \}$
`u := 2 * u` Ass_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}} \wedge \mathtt{i} \geq 1 \wedge \mathtt{i} - 1 \neq \mathtt{n} \}$
$\Longrightarrow$ Cons_P
$\{ P \wedge \mathtt{u} = \mathtt{y} \cdot 2^{\mathtt{i}} \wedge \mathtt{i} \geq 0 \}$

Auch die neue Schleife wird nur logarithmisch oft durchlaufen, damit ergibt sich als Gesamtkomplexität $\mathcal{O}(log(x)^2)$.