Algorithmen und Datenstrukturen II

Arne Schmidt

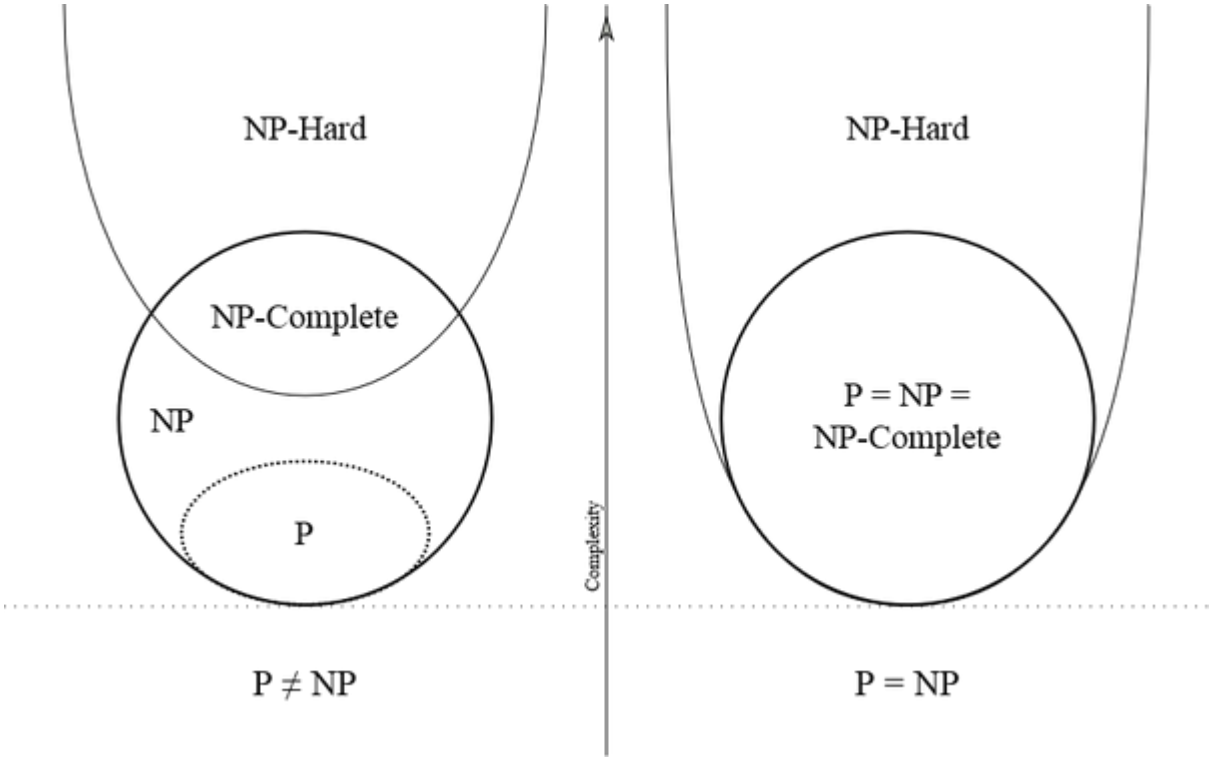
Übung #4

20.06.2018

Übersicht

- Definitionen
- Reduktionen
  - 3Sat -> Knapsack/Subset Sum
  - Subset Sum -> Partition
- Weitere schwere Problem

Definitionen – Ein Bild



https://qph.fs.quoracdn.net/main-qimg-fe98e0dd6f3df4b2a15cb337bebb9fd7

Offene Frage: Gilt P ≠ NP oder P = NP?

Wie zeigt man für ein algorithmisches Problem Π, dass…

- $\boldsymbol{\Pi \in NP}$**:** Für jede Lösung L_I einer Instanz I von Π kann man in polynomieller Zeit prüfen,
  - ob L_I zulässig ist und
  - ob für den Wert P_{L_I} von L_I die Aussage $P_{L_I} \geq K$ gilt, wobei K eine Schranke im Input von I ist.

- $\boldsymbol{\Pi}$ **NP-schwer ist:** Man zeigt, dass man mit einem polynomialzeit Algorithmus für Π jedes NP-schwere Problem lösen kann. (Stichwort: Reduktion!)

- $\boldsymbol{\Pi}$ **NP-vollständig ist:** Man zeigt, dass:
  - Π NP-schwer ist und
  - Π in NP liegt.

Beispiel 0-1-Knapsack

- **Gegeben:** Zahlen $z_1, \dots, z_n, Z, p_1, \dots, p_n, P$
- **Frage:** Existiert eine Lösung $S \subseteq \{1, \dots, n\}$ mit
  - $\sum_{i \in S} z_i \leq Z$ und
  - $\sum_{i \in S} p_i \geq P$?

Liegt 0-1-Knapsack in NP? JA!

1. Ist Rucksack überfüllt? Einfach zu testen!
2. Mindestwert von P erreicht? Auch einfach!

Ist 0-1-Knapsack NP-schwer?

Transformation 3Sat auf Knapsack

Wir haben:

Literale

…

- Klauseln

$$c_1 = (\ell_{1,1}, \ell_{1,2}, \ell_{1,3}), \dots, c_m = (\ell_{m,1}, \ell_{m,2}, \ell_{m,3})$$

- Variablen $x_1, \dots, x_n$
- Variablen tauchen als positive (x) oder negative ($\bar{x}$) Literale in den Klauseln auf.

Wir wählen:

- $b_{x,i} = 1$, falls x in c_i auftaucht; ansonsten 0.
- $\overline{b_{x,i}} = 1$, falls $\bar{x}$ in c_i auftaucht; ansonsten 0.

Transformation 3Sat auf Knapsack

Für die j-te Variable, setze

$$p_{2j} \quad := z_{2j} \quad := 10^{m+n-j} + \sum_{i=1}^{m} b_{j,i} \cdot 10^{m-i}$$

$$p_{2j+1} := z_{2j+1} := 10^{m+n-j} + \sum_{i=1}^{m} \overline{b_{j,i}} \cdot 10^{m-i}$$

Außerdem für die i-te Klausel

$$p_{2n+2i} \quad := z_{2n+2i} := 2 \cdot 10^{m-i}$$

$$p_{2n+2i+1} := z_{2n+2i+1} := 10^{m-i}$$

Wir setzen für die Kapazität und die Wertschranke

$$P := Z := \sum_{j=1}^{n} 10^{m+n-j} + \sum_{i=1}^{m} 4 \cdot 10^{m-i}$$

Transformation 3Sat auf Knapsack

Ein Beispiel!

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

Mit den Formeln erhalten wir:

$$\begin{aligned} z_1 &= 100\ 10 \\ z_2 &= 100\ 01 \\ z_3 &= 010\ 11 \\ z_4 &= 010\ 00 \\ z_5 &= 001\ 01 \\ z_6 &= 001\ 10 \end{aligned}$$

sowie

$$\begin{aligned} z_7 &= 000\ 20 \\ z_8 &= 000\ 10 \\ z_9 &= 000\ 02 \\ z_{10} &= 000\ 01 \end{aligned}$$

Und

$$Z = \underbrace{111}_{\text{Variablen}}\ \underbrace{44}_{\text{Klauseln}}$$

Reduktion von A nach B

- Wandle den Input von A um, sodass er für B dient.
- Alles in polynomieller Laufzeit!
- Aus einer Lösung für B muss eine Lösung für A in polynomieller Laufzeit berechenbar sein.

Konsequenz: Gibt es einen polynomial-zeit Algorithmus für B, gibt es auch einen für A!

Was wissen wir bisher?

Problem	Komplexitätsklasse
3Sat	NP-vollständig
0-1-Knapsack	NP-vollständig
Maximum-Knapsack	NP-schwer
Subset Sum	NP-vollständig
Partition	NP-vollständig
Fractional Knapsack	P
Sortieren (AuD I)	P
Kürzeste Wege (NWA)	P

Schauen wir uns ein paar mehr Reduktionen an!

Subset Sum auf Partition

- Gegeben: $z_1, \dots, z_n, Z$
- Wir müssen Objekte erschaffen, dass $\sum_{i=1}^{m} \widetilde{z_i} = 2Z$.
- Wie wäre es mit:
  - $\widetilde{z_i} := z_i$ für alle $i \in \{1, \dots, n\}$ und
  - $\widetilde{z_{n+1}} := 2Z - \sum_{i=1}^{n} z_i$
- Wir übergeben alle $\widetilde{z_i}$ an das Partition Problem

Nun ist zu zeigen:

Es gibt genau dann

eine Lösung mit $\widetilde{z_1}, \dots, \widetilde{z_{n+1}}$ für Partition,

wenn

es eine Lösung mit $z_1, \dots, z_n, Z$ für Subset Sum gibt

Der Beweis

- Für eine Lösung $\tilde{S}$ für Partition gilt

$$\sum_{i \in \tilde{S}} \widetilde{z_i} = Z = \sum_{i \in \{1,\ldots,n+1\} \setminus \tilde{S}} \widetilde{z_i}$$

- Eine der Mengen unter dem Summenzeichen ist auch eine Lösung für Subset Sum!

- Haben wir eine Lösung S für Subset Sum, so ist diese auch eine Lösung für Partition!

STOP!

Es gibt ein kleines Problem…

$\widetilde{z_{n+1}} := 2Z - \sum_{i=1}^{n} z_i$ kann negativ sein!

Betrachte: $Z' := \sum_{i=1}^{n} z_i - Z$

Nimm diesen Wert statt Z.

Die Reduktion bleibt die gleiche!

Was gibt es noch für Probleme?

Classic Nintendo Games are (Computationally) Hard

Greg Aloupis* Erik D. Demaine† Alan Guo†‡ Giovanni Viglietta§

February 10, 2015

Abstract

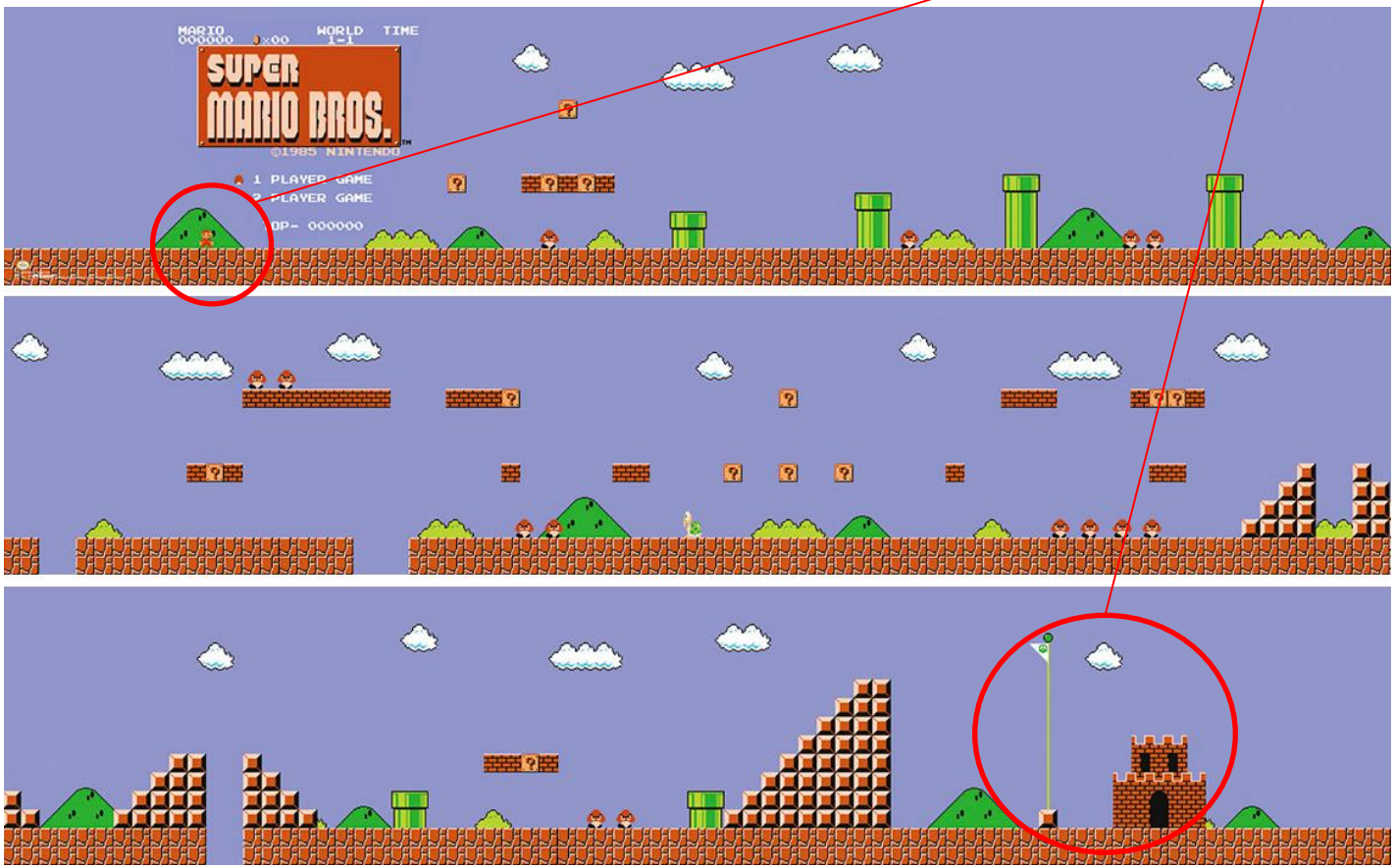
We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon. Our results apply to generalized versions of Super Mario Bros. 1–3, The Lost Levels, and Super Mario World; Donkey Kong Country 1–3; all Legend of Zelda games; all Metroid games; and all Pokémon role-playing games. In addition, we prove PSPACE-completeness of the Donkey Kong Country games and several Legend of Zelda games.

https://arxiv.org/pdf/1203.1895.pdf

Im folgenden

Gegeben: Ein Level eines Spiels

Frage: Gibt es einen Weg vom Start ins Ziel?



https://i.kinja-img.com/gawker-media/image/upload/t_original/rp9crpla17856bynctsm.jpg

Framework

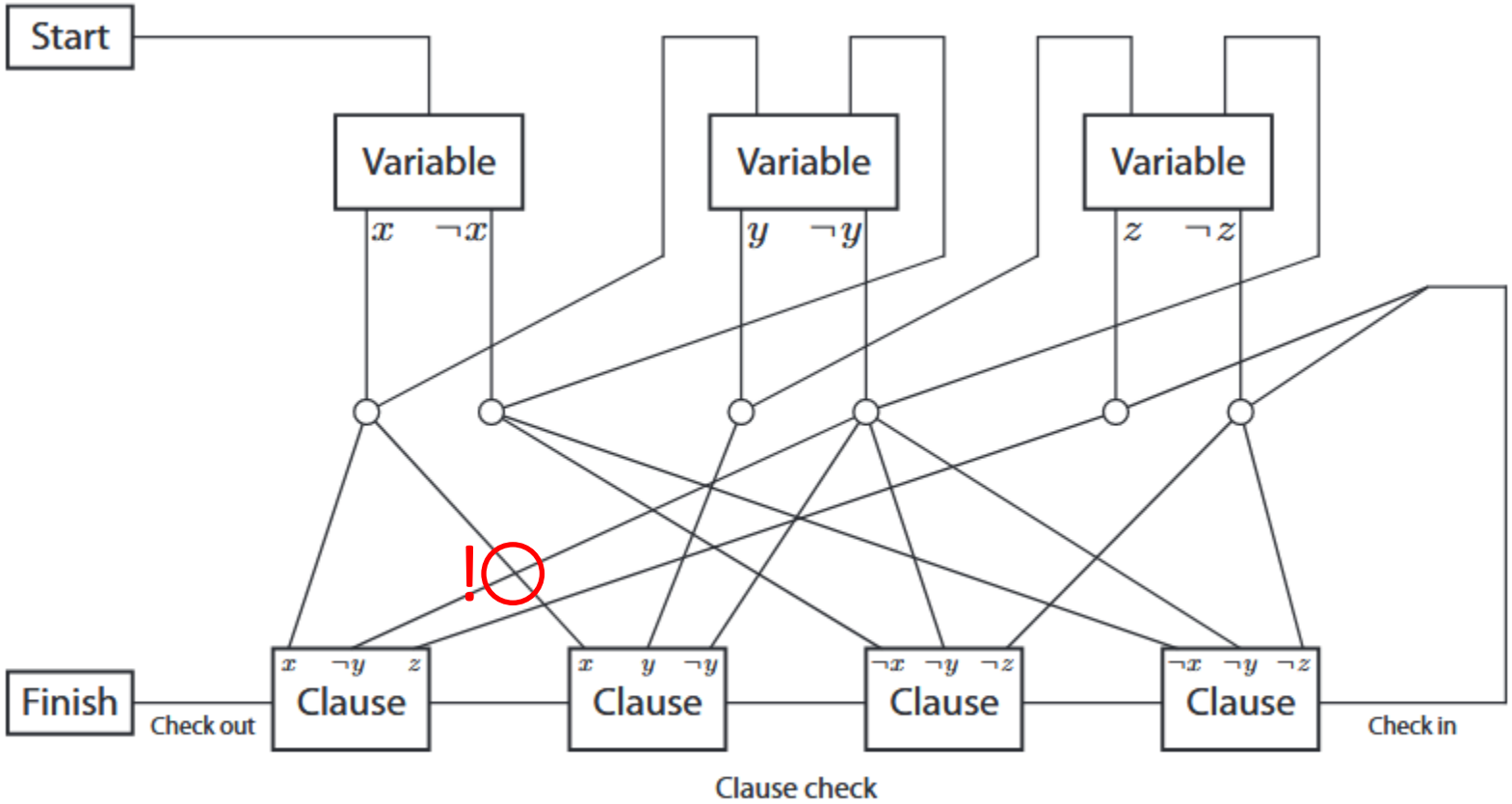


Figure 1: General framework for NP-hardness

Super Mario

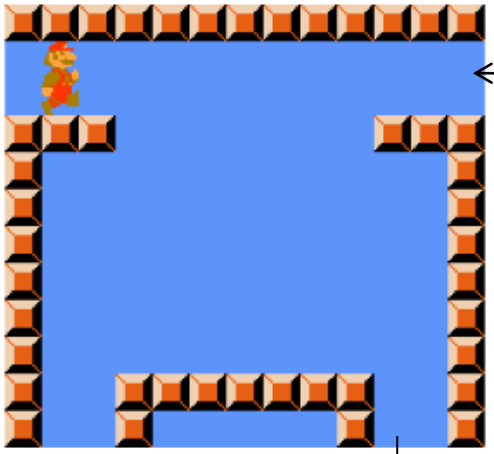
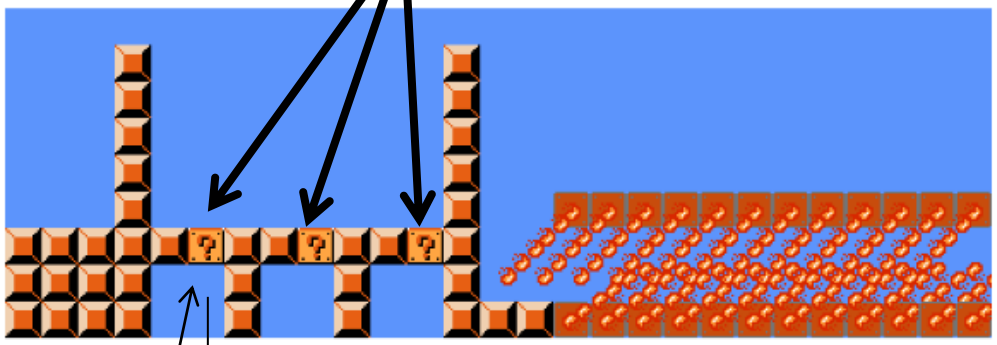


Figure 10: Variable gadget for Super Mario Bros.

Nächste Variable

Enthalten alle einen Stern
→ Unbesiegbarkeit!



Variable true/false setzen

Figure 11: Clause gadget for Super Mario Bros.

Super Mario

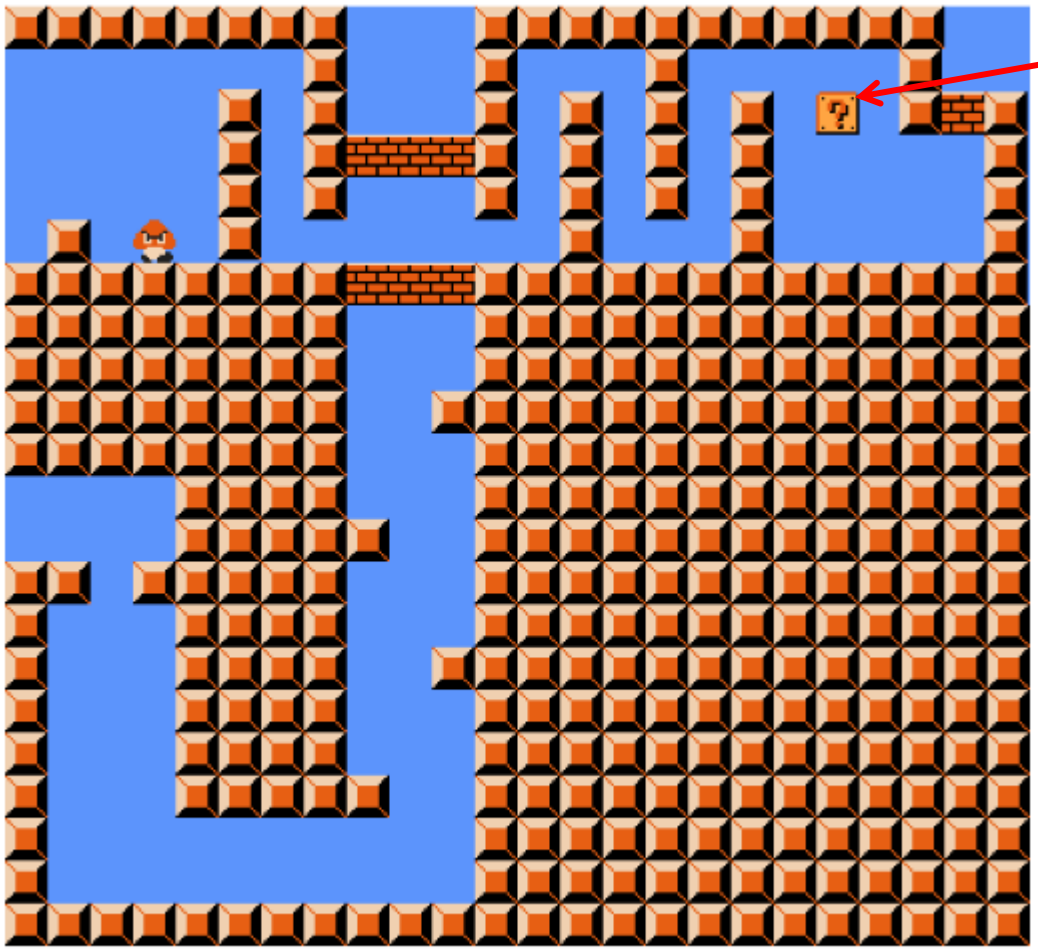


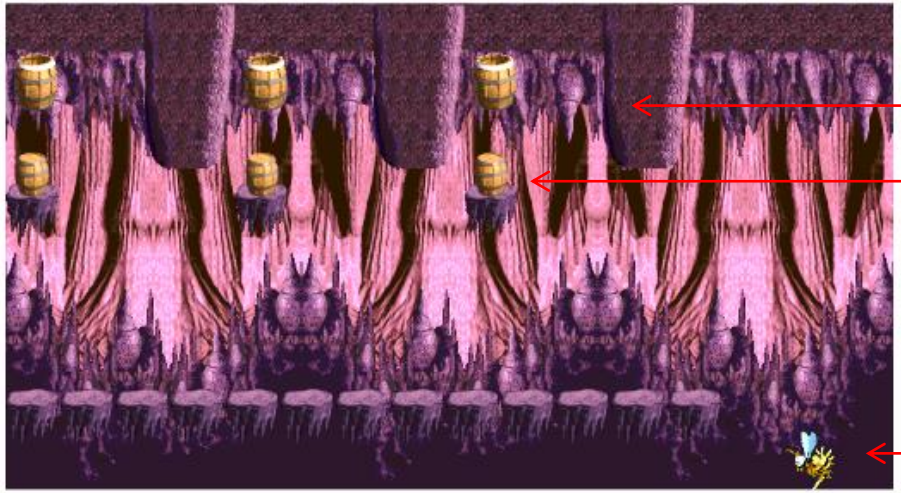
Figure 12: Crossover gadget for Super Mario Bros.

Enthält Pilz -> Mario wird wieder groß!

Links nach rechts:
Nur, wenn Mario klein!

Unten nach oben:
Nur, wenn Mario groß!

Donkey Kong (DK)

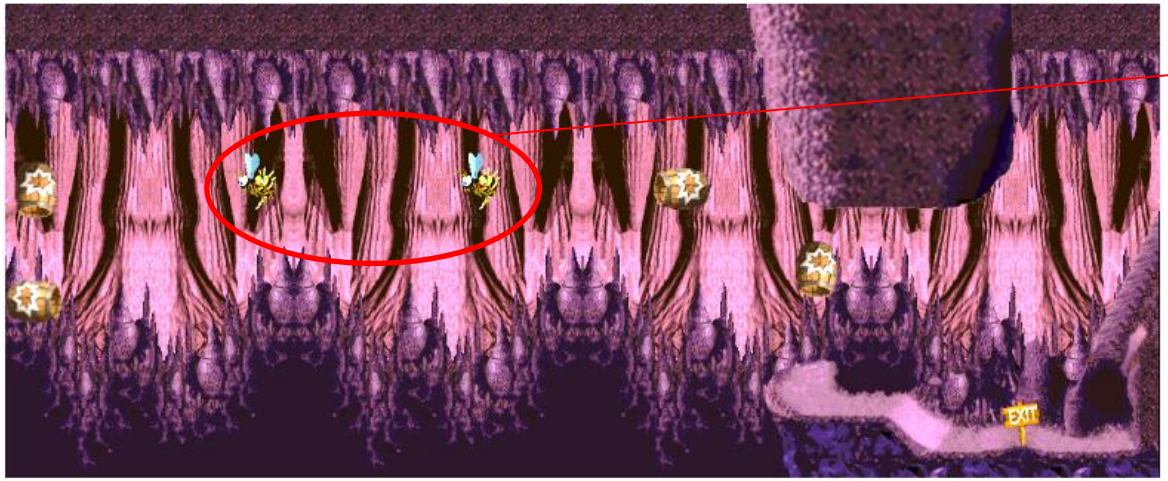


Zurück zur Ebene

Fass tötet Biene

Biene tötet DK

Figure 18: Clause gadget for Donkey Kong



Pro Klausel eine Biene

Figure 19: Clause check for Donkey Kong

Donkey Kong (DK)

Figure 20: Crossover gadget for Donkey Kong

Legend of Zelda

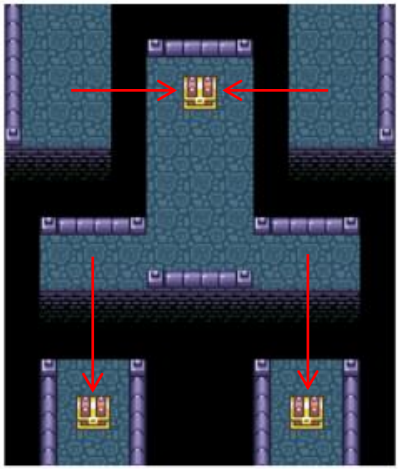


Figure 25: Variable gadget for Zelda

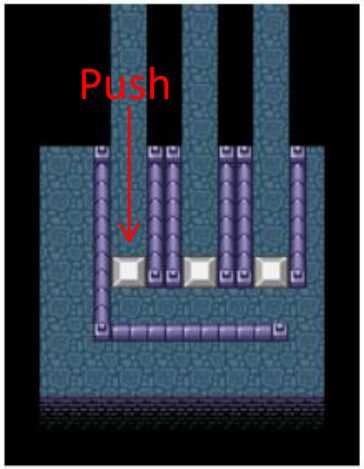
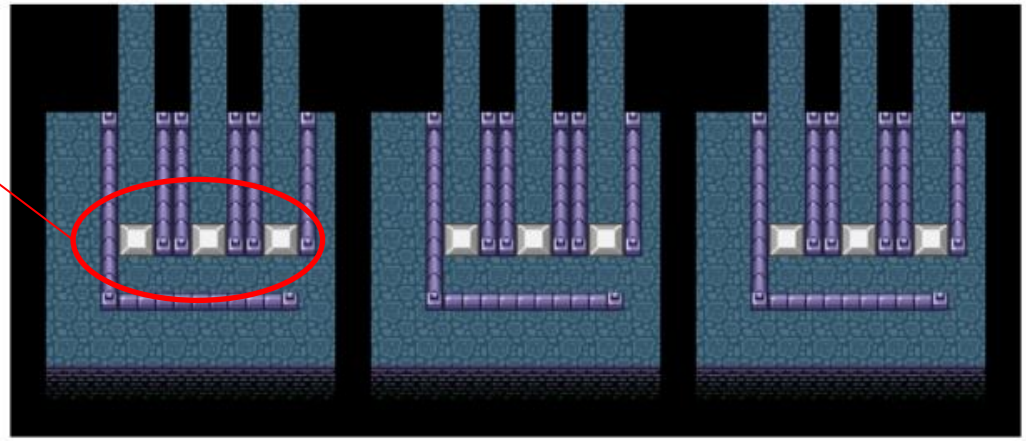


Figure 26: Clause gadget for Zelda

Pro Klausel muss ein Block nach unten verschoben sein

Check out



Check in

Figure 27: Check path for Zelda (with three clauses)

Legend of Zelda

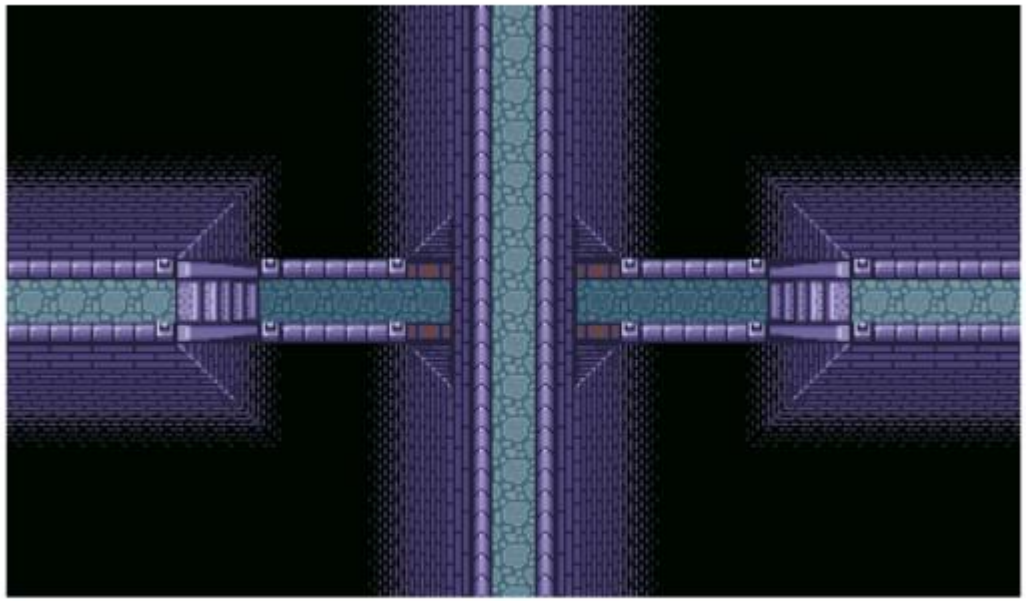


Figure 28: Crossover gadget for Zelda

Pokémon

Wir	Rote Trainer	Blaue Trainer
Nebulak Typ: Geist Bekommt keinen Schaden von *normal* Attacken	Lektroball	Relaxo
Attacke: Explosion (Vernichtet Nebulak)	Greift als erstes mit Explosion (Typ: Normal) an	Greift nicht als erstes an.

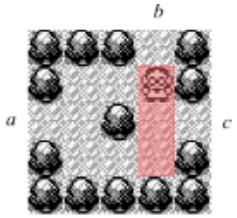


Figure 33: Variable gadget for Pokémon

Deaktiviere mindestens einen roten

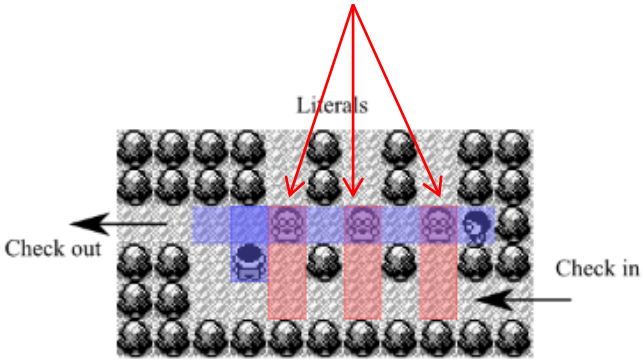


Figure 34: Clause gadget for Pokémon

Pokemon

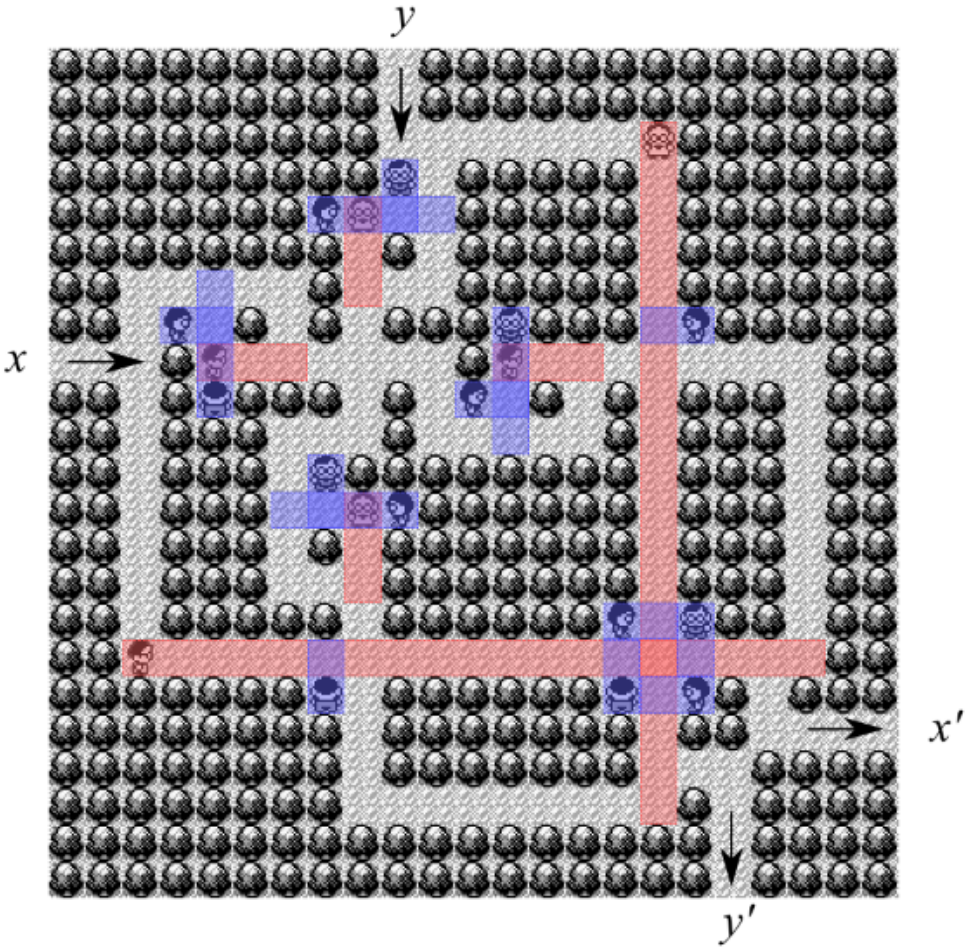


Figure 36: Crossover gadget for Pokémon