Algorithmen & Datenstrukturen

Julian Huber & Matthias Panny

UML - Activity Diagramme

🎯 Lernziele

- Studierende wissen was Activity Diagramme (dt. Aktivitätsdiagramme) sind
- Studierende können Activity Diagramme für simple Probleme erstellen

Activity und Fluss-Diagramme

Bedarf

- Algorithmen und Software sind komplexe Gebilde mit verschiedenen Zuständen
- In der Anfangsphase der Entwicklung muss das Know-How verschiedener Personen mit eingebunden werden

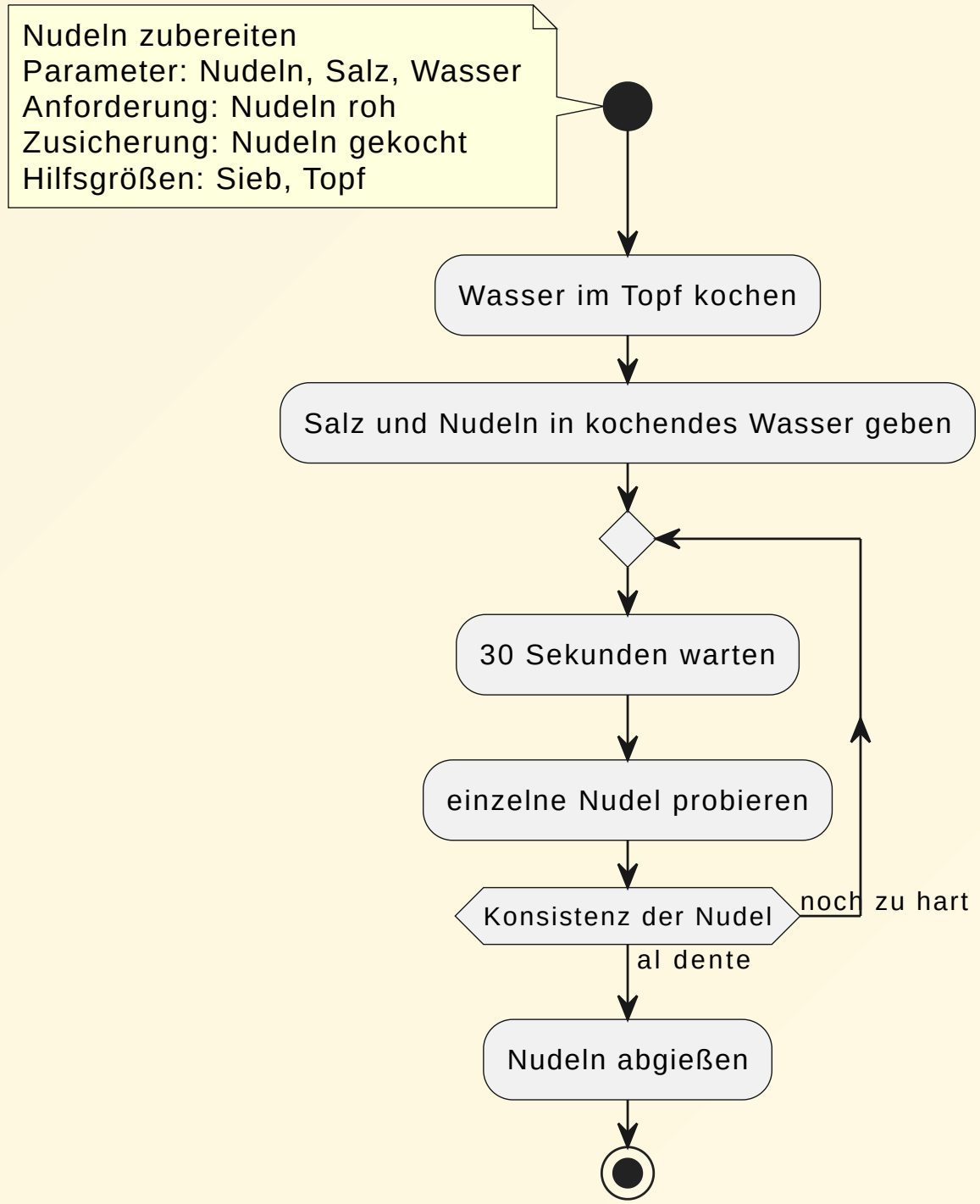
Grafische Darstellung von Programmflüssen

- Es haben sich verschiedene grafische Darstellungsformen entwickelt:
  - Nicht-standardisierte Flussdiagramme
  - Aktivitätsdiagramme der *Unified Modeling Language (UML)* in Version 1 und 2
  - Verwandte Darstellungsformen sind auch Petri-Netze und Zustandsübergangsdiagramme

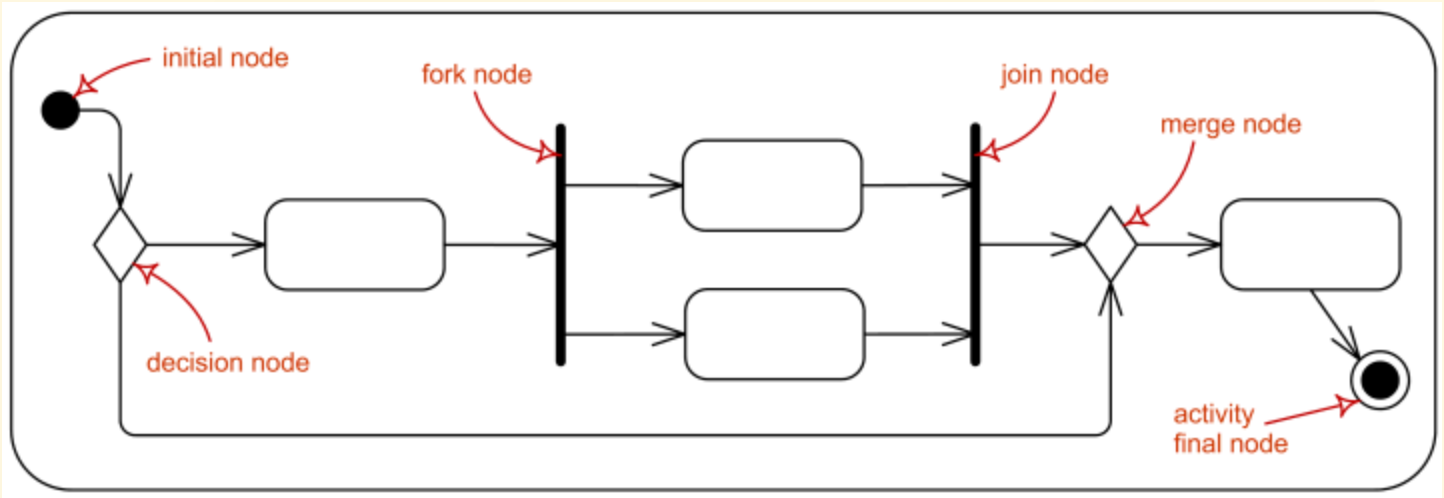
Aktivitätsdiagramm nach UML1

Aktivitätsdiagramm

- dynamische Verhalten eines Software-Systems
- Aktivitäten haben die Struktur eines Graphen
- Knoten dieses Graphen sind Aktionen sowie Punkte, an denen die Flüsse zwischen den Aktionen kontrolliert werden
- Kanten stehen für Daten- und Kontrollflüsse



Aktivitätsdiagramm nach UML1



- Aktivitätsdiagramme bestehen aus Aktivitäten und Kontrollknoten
- Aktivitäten sind Aktionen, die ausgeführt werden
- Kontrollknoten steuern den Fluss der Aktivitäten
  - Decision: Verzweigung basierend auf Entscheidungsalternativen auf den Pfeilen
  - Join: Vereinheitlichung paralleler Prozesse (wartet)
  - Merge: Führt alternative Prozesse wieder zusammen
- ausführliche Beschreibung

Werkzeuge zum Erstellen von Aktivitätsdiagrammen

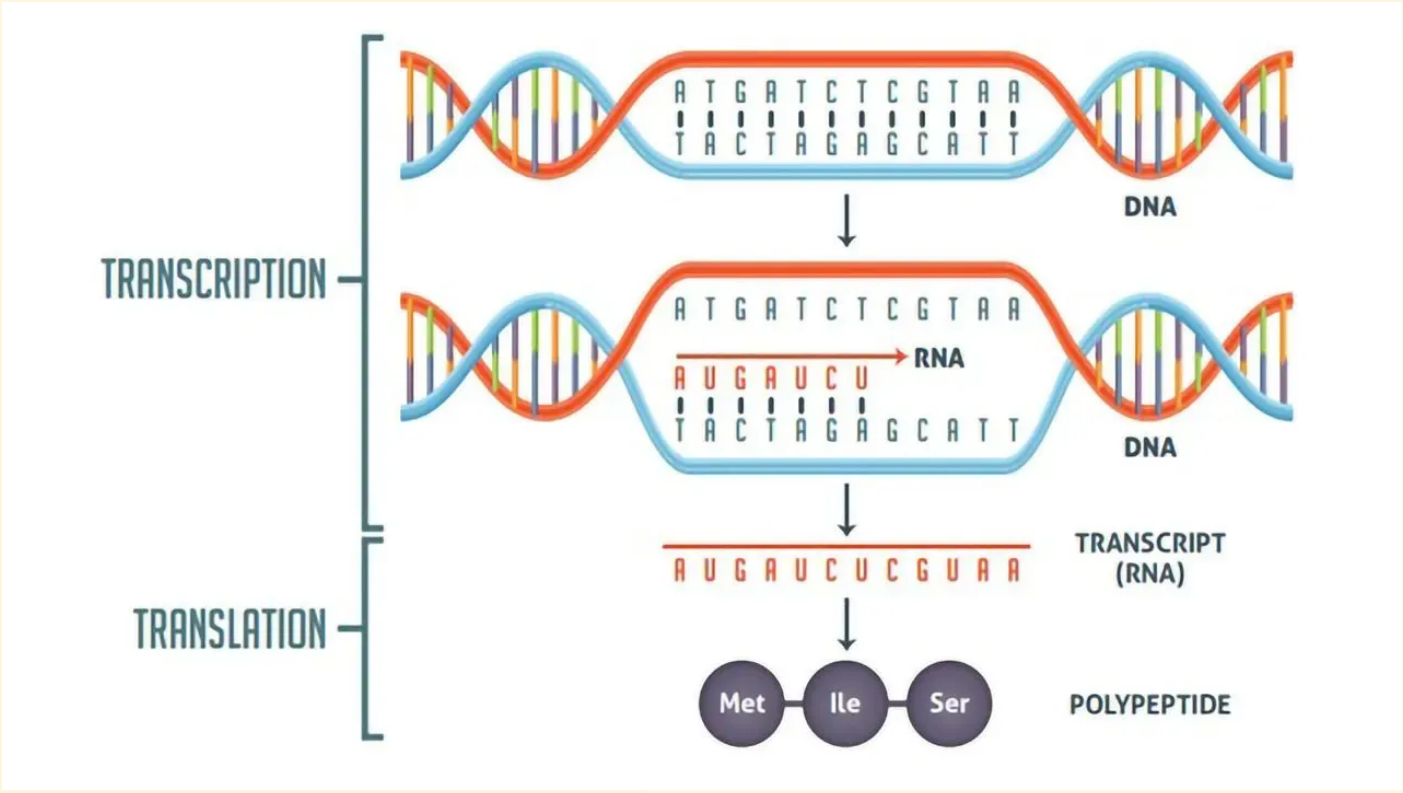
- Diverse Werkzeuge als Online-Tools verfügbar
- Diagrams.net / draw.io → händisch zeichnen
- mermaid.live ohne UML-Konformität

PlantUML

- Auszeichnungssprache für Diagramme

```
@startuml

start
note left
  Nudeln zubereiten
  Parameter: Nudeln, Salz, Wasser
  Anforderung: Nudeln roh
  Zusicherung: Nudeln gekocht
  Hilfsgrößen: Sieb, Topf
end note

:Wasser im Topf kochen;
:Salz und Nudeln in kochendes Wasser geben;

repeat
  :30 Sekunden warten;
  :einzelne Nudel probieren;
repeat while (Konsistenz der Nudel) is (noch zu hart) not (al dente)

:Nudeln abgießen;
stop

@enduml
```

Beispiel - Reverse Complement eines DNA-Strangs



- der genetische Code einer Zelle ist als doppelsträngige Helix in der DNA gespeichert
- Dabei ist die Information der beiden Stränge redundant gespeichert, da sich immer nur zwei bestimmte Basenpaare von vier gegenüber liegen können (`A-T` bzw. `C-G`)
- Zudem haben die beiden Stränge eine gegensätzliche Lese-Richtung (5'-3')

Beispiel - Reverse Complement eines DNA-Strangs

- Ein häufiges Problem der Bio-Informatik ist es das Reverse-Complement einer Nukleotid-Folge zu finden
- Complement bedeutet, dass die Basen `A` und `T` sowie `C` und `G` ersetzt werden um den zweiten Strang zu erhalten
- Reverse bedeutet, dass die Reihenfolge der Basen umgekehrt wird, um die Lese-Richtung zu erhalten

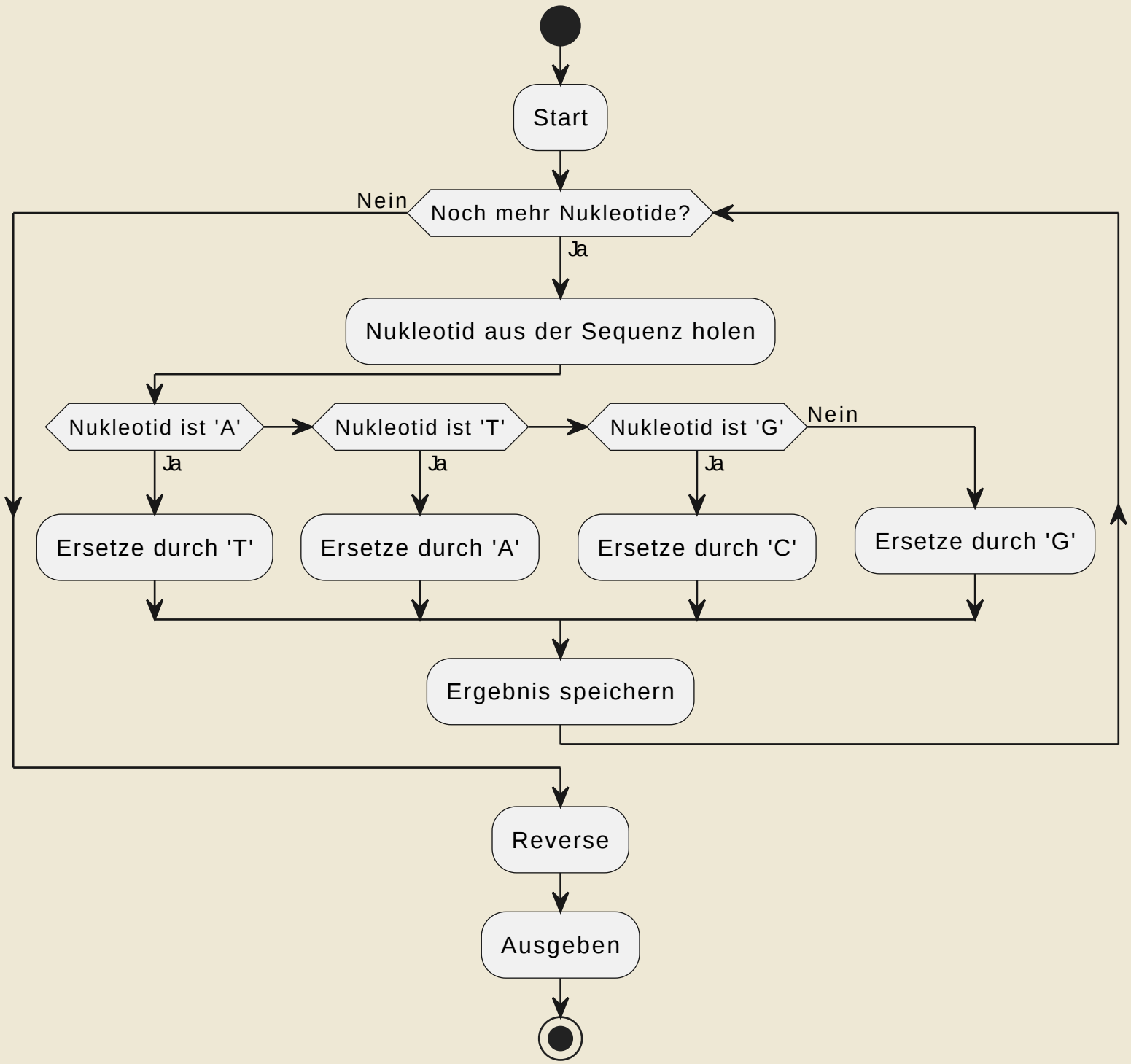
```
>Sample Sequence
ATGATCTCGTAA

>Complement
TACTAGAGCATT

>Reverse Complement
TTACGAGATCAT
```

✍️ Aufgabe - Reverse Complement eines DNA-Strangs

- Klonen Sie das Repository `example_reverse_complement`
- Im `README.md` finden Sie eine Beschreibung des Problems und der Aufgabenstellung → Arbeiten Sie diese Schritt für Schritt durch
- Begonnen wird mit dem Zeichnen eines UML-Aktivitätsdiagramm zur Lösung dieses Problems
  → gehen sie davon aus, dass Sie nur binäre Fallunterscheidungen treffen können

Beispiel - Reverse Complement eines DNA-Strangs

Musterlösung für das Aktivitätsdiagramm



Beispiel - Reverse Complement eines DNA-Strangs

- Auch für dieses Beispiel sind Musterlösungen vorhanden

Musterlösung - `reverse_complement.py`

- Diverse unterschiedliche Implementierungen
  - ohne Fehlerbehandlung
  - mit Fehlerbehandlung
  - mit `list`
  - mit `dict`

Musterlösung - `test_reverse_complement.py`

- Unit Test für die Implementierungen ohne Fehlerbehandlung
- Unit Test für die Implementierungen ohne Fehlerbehandlung
- Ausführen im Terminal:
  `python -m unittest .\test_reverse_complement.py` bzw. den ganzen `test` Ordner über `python -m unittest discover test`

Beispiel - Reverse Complement eines DNA-Strangs

- Im 4. Schritt der Aufgabenstellung wird die Erweiterung der Implementierung um Fehlerbehandlung gefordert

Problem

- Fehlerbehandlung bei unbekannten Buchstanden

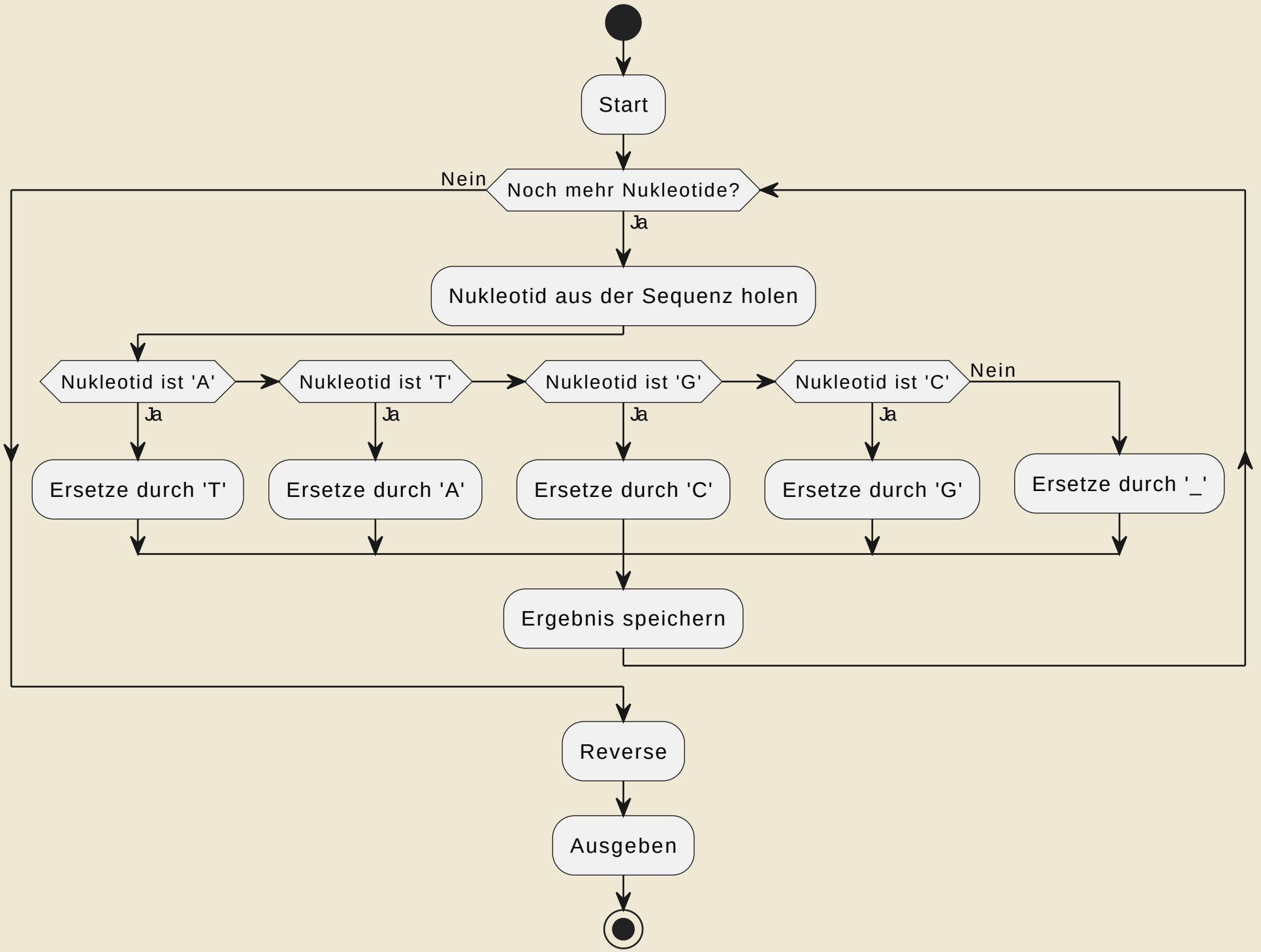
```
DNA Sequence:          ATGATCTCGTAA
Reverse Complement:    TTACGAGATCAT

DNA Sequence:          AXGATCTCGTAA
Reverse Complement:    TTACGAGATCT
```

Lösung

- Es soll jedes ungekannte Nukleotid durch _ ersetzt werden
- Welche alternativen Lösungsansätze fallen Ihnen ein?

Beispiel - Reverse Complement eines DNA-Strangs

Musterlösung



Beispiel: Übersetzer für längere Alphabete

- Wie gehen wir damit um, dass es auch längere Mappings (also mehr als fünf Möglichkeiten) geben kann?

Problem

- Wir wollen nicht für jedes Mapping eine neues Paar an `if`-Bedingung implementieren

Lösung

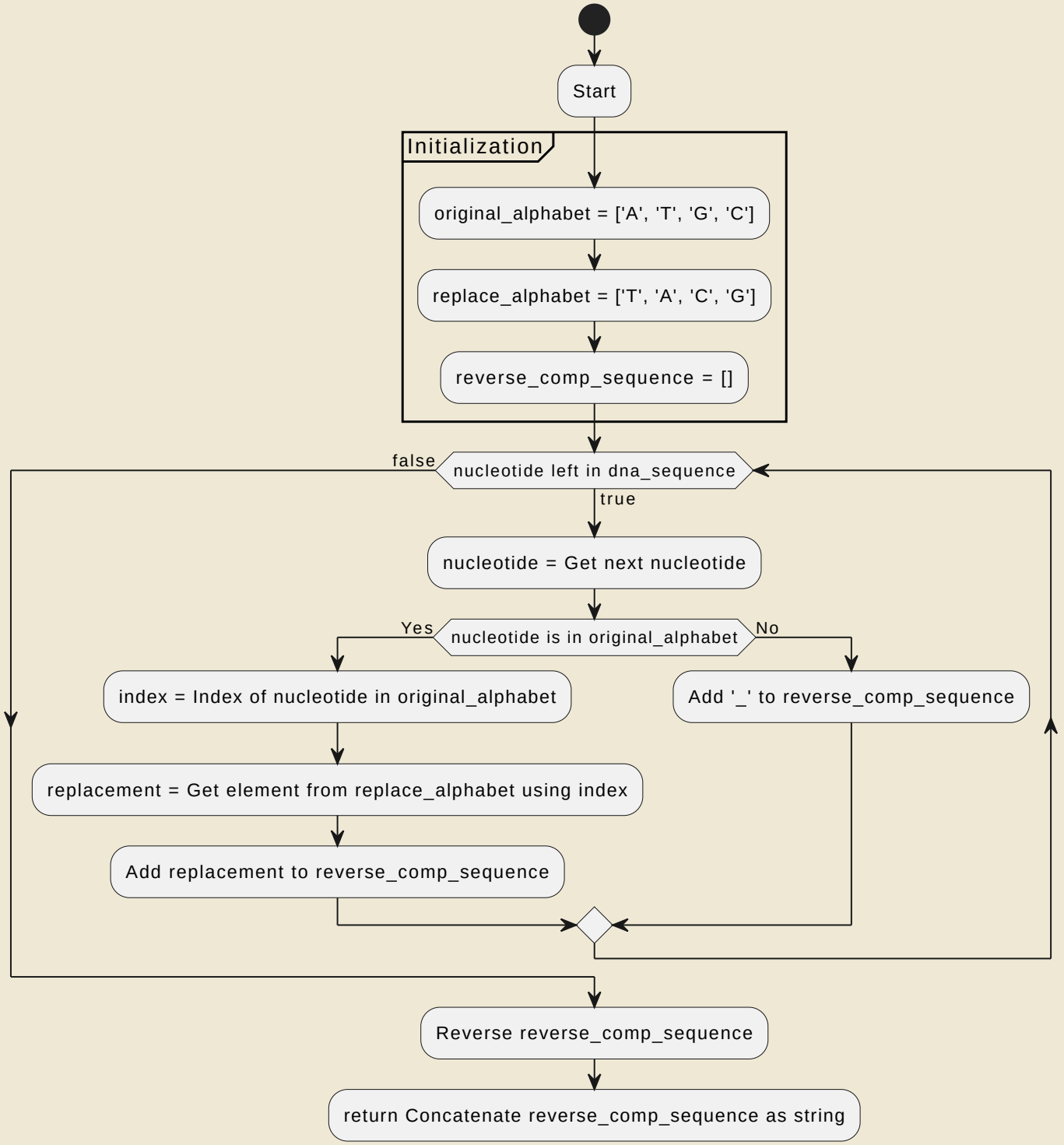
- Stattdessen können wir ein Mapping über zwei Listen mit gemeinsamen Index anlegen

```
original_alphabet = ['A', 'T', 'G', 'C']
replace_alphabet  = ['T', 'A', 'C', 'G']
```

- ⚠️: Auch dieses Vorgehen hat Nachteile über welche wir zu späterem Zeitpunkt sprechen

Beispiel: Übersetzer für längere Alphabete

Musterlösung



✍️ Aufgabe

- Passen Sie die Implementierung von `reverse_complement(dna_sequence)` an, da dass zwei Listen mit Alphabete verwendet werden und nicht erkannte Nukleotide durch `_` ersetzt werden
- *Optional:* Passen Sie den unit-Test an, damit diese Funktionalität auch getestet wird und führen Sie den unit-Test erneut aus
- Welche alternativen Implementierungen fallen Ihnen ein?