Spiele

Theorie eines Spiels

mathematisch: k-Spieler Spiel ist Graph $G = (V, E)$ wobei V partitioniert in $V_1, \ldots, V_k$

Knoten $v \in V$ heissen *Konfigurationen*

oft gegeben: $v_0 \in V$ Startkonfiguration (Anfangszustand)

Folgen

Def.: Sei M Menge, M^* bezeichnet die Menge aller, endlichen Folgen von Elementen aus M.

M^+ ist die Menge alle nicht-leeren, endlichen Folgen über M.

Bem.: $M^+ = MM^*$

Def.: (Präfixordnung)
Geg. $f, g \in M^*$, dann ist $f \preceq g$, falls $\exists h \in M^*, fh = g$

Partien

hier: nur endliche Partien

$\pi \in V^+$ heisst *Partie*, falls gilt:

- $\pi = v_0 v_1 \ldots v_n$ für ein $n \in \mathbb{N}$
- für alle $i = 0, \ldots, n-1$: $(v_i, v_{i+1}) \in E$

Entstehen einer Partie

induktiv:

Start: $\pi := v_0$

nächster Schritt: gegeben $\pi = v_0 \dots v_n$,

1. $v_n \in V_i$ für ein $i \in \{1, \dots, k\}$
2. Spieler i wählt $v_{n+1} \in V$ mit $(v_n, v_{n+1}) \in E$
3. Partie wird $\pi := v_0 \dots v_n v_{n+1}$

Gewinnbedingungen

eine Gewinnbedingung ist eine partielle Abbildung $\mathcal{W} : V^+ \to \{1, \ldots, k\}$ mit

- π keine Partie : $\mathcal{W}(\pi)$ undefiniert (warum nicht $\Leftrightarrow$?)
- für jede Partie π existiert Partie π' mit $\pi \preceq \pi'$ und $\mathcal{W}(\pi')$ is definiert
- falls $\pi \preceq \pi'$ und $\mathcal{W}(\pi) = i$, dann auch $\mathcal{W}(\pi') = i$

ist $\mathcal{W}(\pi) = i$, dann sagen wir, dass Spieler i die Partie π gewinnt

Erreichbarkeitsspiele

spezielle, einfache Art von Spielen

Gewinnbedingung partielle Abbildung $\mathcal{W} : V \to \{1, \ldots, k\}$

Spieler i gewinnt Partie $\pi = v_0 \ldots v_n$ falls $\mathcal{W}(v_n) = i$

Gewinnstrategien

eine Gewinnstrategie für Spieler i ist eine partielle Abb. $\sigma : V^+ \to V$, mit

- falls $\pi = v_0 \ldots v_n$ Partie und $v_n \in V_i$, dann ist $\sigma(\pi)$ definiert und $(v_n, \sigma(\pi)) \in E$
- Spieler i gewinnt jede Partie, solange er die Züge in σ macht

Thm.: In jedem Spiel hat höchstens 1 Spieler eine Gewinnstrategie.

Beispiel

Auswertungsproblem für boolesche Ausdrücke

Geg. Term f über $0, 1, \wedge, \vee$,
ist $f \equiv 1$?

Konfigurationen: Subterme
Startkonf.: f
Übergänge und Gewinnbedingungen:

- Spieler 1 zieht von $g \wedge h$ nach g oder h, gewinnt in 0
- Spieler 2 zieht von $g \vee h$ nach g oder h, gewinnt in 1

Thm.: $f \equiv 1$ gdw. Spieler 2 eine Gewinnstrategie hat

Determiniertheit

Def.: Ein Spiel heisst *determiniert*, falls genau 1 Spieler eine Gewinnstrategie hat.

Termauswertungsspiel z.B. ist determiniert.

Der Satz von Zermelo

Thm.: (Zermelo, 1913)
Jedes 2-Spieler Erreichbarkeitsspiel ist determiniert.

Bem.:

- Motivation war Schachspiel
- gilt nicht für mehr als 2 Spieler

ab jetzt nur noch 2-Spieler-Spiele

Bedeutung der Determiniertheit

Beweis der Determiniertheit kann Strategie explizit konstruieren

Bsp.: Termauswertungsspiel, Beweis der Determiniertheit
Induktion über Termstruktur

$f = 0$ oder $f = 1$: Spieler 2, bzw. 1 hat triviale Gewinnstrategie

$f = g \vee h$: Spieler 2 muss g oder h wählen
nach Voraussetzung Spiele für g und h determiniert
Spieler 2 hat Gewinnstrategie für f, falls er eine für g oder h hat
falls nicht, so hat Spieler 1 eine Gewinnstrategie für f

$f = g \wedge h$: analog

Explizite vs. implizite Darstellung

explizite Darstellung der Strategie für Spieler 2:

Baum der Subterme,
$\vee$-Knoten hat einen Nachfolger
$\wedge$-Knoten hat zwei Nachfolger

implizite Darstellung:

Spieler 2 wählt den Subterm, der zu 1 auswertet

Problem: implizite Darstellung zwar kompakt, aber unbrauchbar zur Lösung des Auswertungsproblems

Positionale Determiniertheit

Def.: Eine *positionale Gewinnstrategie* ist eine Gewinnstrategie vom Typ: $V \to V$

Idee: aktueller Zug hängt nur von aktueller Konfiguration ab, nicht von bisherigem Verlauf der Partie

andere Bezeichnung: *history-free*, *memory-less*

Def.: positionale Determiniertheit = Determiniertheit mit positionalen Gewinnstrategien

Thm.: Jedes 2-Spieler Erreichbarkeitsspiel ist positional determiniert.

Bedeutung der positionalen Determiniertheit

positionale Gewinnstrategien lassen sich explizit als Graph statt als Baum speichern

Bsp.: $f(x) = x \vee x$, $g(x) = x \wedge x$

$$\underbrace{g(f(g(f(g(f(\ldots g}_{n \text{ mal}}(1)\ldots))))))$$

Strategie als Baum hat Grösse $O(2^n)$
Strategie als Graph hat Grösse $O(n)$

Spiele mit Unentschieden

s. originale Arbeit von Zermelo

Modellierung genauso, aber Gewinnbedingung $\mathcal{W} : V^+ \to \{1, 2, U\}$

jetzt “gewinnen” und “verlieren” nicht mehr dual

ersetze “gewinnen” durch “nicht verlieren”, dann

Determiniertheit bleibt erhalten, da es keinen Spieler U gibt

Spiele mit mehr als 2 Spielern

Reduktion auf 2-Spieler Modell:

“Wenn ich verlieren, dann ist es mir egal, wer gewinnt.”

aus Sicht des Einzelnen jeweils als 2-Spieler-Spiel behandeln

Würmer Versenken

Fragen:

- Ist “Würmer Versenken” ein 2-Spieler-Spiel im obigen Sinne?
- Ist es ein Erreichbarkeitsspiel?

Konf.: SpielVariante + . . .

Gewinnbed.: i gewinnt in Konf. ohne Würmer für i

Würmer Versenken

in jeder 2-Spieler-Version von “Würmer Versenken” hat ein Spieler eine Gewinnstrategie

genauer: eine Strategie, nicht zu verlieren

wozu Spiel noch spielen lassen?

Komplexität

Bsp.

- Spielbrett der Grösse 20 x 20,
- 50% freie Felder
- je 5 Würmer der Länge 1

Anzahl verschiedener Konfigurationen ca.

$$5.657.653.085.871.350.000$$

Problem

implizite Strategie-Repräsentation nicht konstruktiv

explizite Strategie-Repräsentation nicht durchführbar wegen Komplexität

Lösung: Heuristiken

Def.: Heuristik = Näherungsverfahren ohne Garantie auf Erfolg

Aufgabenstellung

Programmieren eines automatischen Spielers (einer Strategie, einer Heuristik)

nicht akzeptabel: trivialer Spieler, der z.B. nur zufällige Züge macht

Implementierung dessen zu Testzwecken jedoch empfohlen

Stoff zu Strategien, Heuristiken: nur Vorschläge, keine Vorschriften

Bewertungsfunktion

eine Bewertungsfunktion γ misst die Güte einer Konfiguration, also z.B.
$\gamma : V \to \mathbb{N}$

Bsp.: $\gamma(v) = \sum_{\text{eigene Würmer w}} distanz(w, \text{Loch})$

Bsp.:
$\gamma_1(v) =$ “wie schnell komme ich zum Loch”
$\gamma_2(v) =$ “wie gut kann ich blockieren”
$\gamma(v) = c_1 \cdot \gamma_1(v) + c_2 \cdot \gamma_2(v)$

Greedy-Algorithmus

greedy = tue das momentan beste

berechne Bewertungsfunktion für alle möglichen Folgekonfigurationen
mache Zug zu derjenigen mit bester Bewertung

+ leicht zu implementieren
+ kaum Speicherverbrauch
- interessiert sich nicht für Züge des Gegners
- greedy-Strategien oft nicht optimal
- verschwendet Rechenzeit

Approximation

berechne nicht vollständigen Graph der Konfigurationen, sondern nur Teil davon, abhängig von verfügbarer

- Rechenzeit
- Speicherplatz

\+ berücksichtigt Züge des Gegners

\+ nutzt Rechenzeit besser aus

\- garantiert nicht, keinen schlechten Zug zu machen