COMPLEXITY OF PATH FORMING GAMES

Hans L. Bodlaender

RUU-CS-89-29
December 1989

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

COMPLEXITY OF PATH FORMING GAMES

Hans L. Bodlaender

Technical Report RUU-CS-89-29
December 1989

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

COMPLEXITY OF PATH FORMING GAMES*

Hans L. Bodlaender
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, the Netherlands

Abstract

For a number of two player games where players alternately choose the next vertex of a simple or elementary path in a graph, we consider the problem to determine whether for a given game instance there is a winning strategy for the first player. We show several of these problems to be PSPACE-complete. In some special cases, we obtain polynomial time algorithms, based upon graph rewriting or an intricate form of dynamic programming, e.g. we show GENERALIZED GEOGRAPHY and some other PSPACE-complete problems to be linear time solvable on graphs with constant bounded treewidth.

1 Introduction.

Games are not only a popular pastime, but they can also serve as a model for several different phenomena, e.g.

- conflicts between parties with different interests (e.g. different companies that operate on the same market).
- fault-tolerance. Here the erroneous behavior of a system is modeled by assuming that the system uses an intelligent strategy to prevent us from reaching our goal. If we are able to deal with this type of error, we are also able to deal with all weaker types of errors.
- worst-case complexity of algorithms. See e.g. [31].
- complexity theory. For instance, the definition of the alternating Turing machine (which is one of the standard models for parallel computation) can be stated in terms of a game. (See [24].)

*This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM).

In this paper we restrict ourselves to games with 2 players that have full information. We concentrate on the question: given a certain game, how hard is it to determine whether there is a winning strategy for the first (or second) player. With the notion of "game", we usually mean a class of actual games (instances), where each game (instance) is distinguished from others in the class only by its starting position and playing area, but uses the same type of moves. With the complexity of a game (class), we denote the complexity of the following problem: given a game (instance) from this class, does player 1 have a winning strategy in this game (instance)? Throughout this paper we use the name of a game to denote this problem for that game.

For several problems of this type, PSPACE-completeness and EXPTIME-completeness results have been obtained, as well as for well-known and often played games, like chess, go, and checkers [20, 22, 29], as well as for more abstract games [1, 14, 18, 21, 33, 36, 38]. For an overview, see e.g. [23, 24].

In this paper we consider several abstract games on graphs, that have "forming paths in a graph" as a common theme. We consider two already previously studied games, both known under the name GENERALIZED GEOGRAPHY, and several new games: TRON (a generalization of a popular video-game, based upon the Walt-Disney movie of the same name), the HAMILTONIAN CIRCUIT CONSTRUCTION GAME, the HAMILTONIAN PATH CONSTRUCTION GAME, the SIMPLE PATH CONSTRUCTION GAME, the ELEMENTARY PATH CONSTRUCTION GAME, and variants, where the starting vertices are not specified.

For most of these games we prove the corresponding decision-problem to be PSPACE-complete (under logarithmic space reductions). For a few of these cases we have also a PSPACE-completeness result for the variant with undirected graphs. These results are presented in Section 3.

A huge amount of research has been done on the complexity of NP-complete graph problems, when restricted to special classes of graphs (see e.g. [25]). Very little however is known on the complexity of PSPACE-complete graph problems, when restricted to special classes of graphs. In this paper we give some interesting results in this area. We use as main techniques: graph rewriting and an intricate form of dynamic programming. Among others, we show that (VERTEX) GENERALIZED GEOGRAPHY and some other PSPACE-complete problems are linear time solvable on graphs with bounded treewidth. This is the first known example of a PSPACE-complete problem, solvable in polynomial time when restricted to graphs with bounded treewidth, where previously only such results were known for problems, known to be NP-complete or NP-hard [4, 6, 9, 16, 17, 26, 37, 39]. These results are presented in Section 4. Some final comments are made in Section 5.

2 Definitions.

In this section we give most definitions that are needed in this paper. In Section 2.1 we give the definitions of graph-theoretic notions; in Section 2.2 we give the definitions of the considered games, and of a PSPACE-complete logic problem, used in our PSPACE-completeness proofs.

2.1 Graph-theoretic definitions.

A path in a graph $G = (V, E)$ is called *simple*, if no vertex appears more than once on the path. It is called *elementary*, if no edge $e \in E$ is used more than once by the path. The subgraph of $G = (V, E)$, induced by $W \subseteq V$, is denoted by $G[W] = (W, \{(v, w) \in E \mid v, w \in W\})$.

Next we give the definitions of two special classes of graphs: the cacti, and the graphs with treewidth $\leq k$.

Definition.
An undirected graph $G = (V, E)$ is a cactus (graph), if and only if every edge $e \in E$ belongs to at most one simple cycle in G.

In other words, a graph $G = (V, E)$ is a cactus, if and only if each connected component of G either is a single edge or a cycle without chords.

Definition.
Let $G = (V, E)$ be a directed or undirected graph. A tree-decomposition of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V, and T a tree, such that

- $\bigcup_{i \in I} X_i = V$.
- $\forall (v, w) \in E : \exists i \in I : v \in X_i \wedge w \in X_i$.
- $\forall i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The treewidth of a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of G is the minimum treewidth over all possible tree-decompositions of G.
The problem to determine the treewidth of a graph is NP-complete [3]. However, for fixed k, one can determine in $\mathcal{O}(n^2)$ time whether the treewidth of a given graph is $\leq k$, and if so, find a tree-decomposition with treewidth $\leq k$ [11, 32]. An algorithm, using $\mathcal{O}(n^{k+2})$ time, but with a much better constant factor can be found in [3]. A very recent approach, which may lead to more practical $\mathcal{O}(n^2)$ algorithms can be found in [19].

Several well-studied classes of graphs have the property that there is a constant upper bound for the treewidth of graphs in the class. For instance, cacti and series-parallel graphs have treewidth ≤ 2, Halin graphs have treewidth 3, almost-trees with parameter k have treewidth $\leq k+1$, k-outerplanar graphs have treewidth $\leq 3k-1$. Also, if there is a fixed upper bound on the bandwidth, cutwidth, search number, vertex separation number of graphs in a class, or if the graphs in the class are interval graphs or chordal graphs with a fixed upper bound on the maximum clique size, then there is also a fixed upper bound on the treewidth of the graphs in the class. (See [8, 13, 37, 39].)

There are several equivalent characterizations of the class of graphs with treewidth $\leq k$. For instance, a graph is a partial k-tree, if and only if its treewidth is at most k. (See [2, 37]). Note also that each class of graphs that can be defined recursively in terms of rules of compositions with the method of Bern, Lawler and Wong [7] has a constant upper bound on the treewidth of the graphs in the class.

So, a polynomial algorithm for a certain problem for graphs with bounded treewidth directly implies polynomial algorithms for that problem for a large number of other classes of graphs.

2.2 Definitions of games and logical problems.

In this section we give definitions of some logical problems, used in our PSPACE-completeness proofs, and of the games we consider in this paper. First we give the definition of a well-known logical problem.

QUANTIFIED 3-SATISFIABILITY
Instance: Set $U = \{u_1, u_2, \ldots, u_n\}$ of variables, well-formed quantified Boolean formula $F = (Q_1 u_1)(Q_2 u_2)\ldots(Q_n u_n)E$, where E is a boolean expression in conjunctive normal form with three literals per clause, and each Q_i is either $\forall$ or $\exists$.
Question: Is F true?

QUANTIFIED 3-SATISFIABILITY is PSPACE-complete. One may also assume that the quantifiers alternate, i.e., that $Q_i = \forall$, if i is even, and $Q_i = \exists$, if i is odd, or that $Q_i = \exists$, if i is even, and $Q_i = \forall$, is i is odd. (See [23, 35]).

Next we introduce several games on graphs. We start with two slightly different games, which are both known under the name GENERALIZED GEOGRAPHY. To distinguish the variants, we call them VERTEX GENERALIZED GEOGRAPHY and EDGE GENERALIZED GEOGRAPHY. Both games are played on a directed graph $G = (V, E)$, given with a starting vertex s. In the VERTEX GENERALIZED GEOGRAPHY game, players alternately choose a vertex. The first chosen vertex must be s, and each subsequently chosen vertex must have an incoming edge with the last chosen vertex as its other endpoint. Players may not choose a vertex that has been chosen before. The first player that is unable to move loses the game.

In the EDGE GENERALIZED GEOGRAPHY game, players alternate choosing an edge that has not been chosen before, starting with an edge that has its tail at the vertex that was the head of the previous chosen edge. Again, the first player unable to move loses the game.

Thus, in VERTEX GENERALIZED GEOGRAPHY, players alternately choose the next vertex of a simple path in G, and in EDGE GENERALIZED GEOGRAPHY, players alternately choose the next edge of an elementary path.

Both games are PSPACE-complete. EDGE GENERALIZED GEOGRAPHY was proven to be PSPACE-complete by Schaefer in [36]. In [29] it was proved that VERTEX GENERALIZED GEOGRAPHY is PSPACE-complete for planar, bipartite graphs, that have no vertices with in- or out-degree exceeding 2 or with degree exceeding 3.

In this paper we consider also the following variants of these games:

- SIMPLE PATH CONSTRUCTION GAME. This game is played as VERTEX GENERALIZED GEOGRAPHY, but with the following difference: the instance also contains an integer $k \leq |V|$. Player 1 wins, if and only if the game ends with a path, containing at least k vertices.

- ELEMENTARY PATH CONSTRUCTION GAME. This game is played as EDGE GENERALIZED GEOGRAPHY, but now the instance contains an integer $k \leq |E|$, and player 1 wins, if and only if the game ends with a path, containing at least k edges.

- HAMILTONIAN PATH CONSTRUCTION GAME is the special case of SIMPLE PATH CONSTRUCTION GAME with $k = |V|$. In other words, player 1 wins, if and only if the game ends when all vertices have been visited.

- HAMILTONIAN CIRCUIT CONSTRUCTION GAME. This is similar to the HAMILTONIAN PATH CONSTRUCTION GAME, but now player 1 wins, if and only if the game ends when all vertices have been visited and there is an edge from the last visited vertex to the first visited vertex.

- Variants *without specified starting vertex.* These are similar to the original games, but now player 1 is free to choose whatever vertex or edge he/she wants as starting vertex.

We also consider a game, that we call TRON. An instance of TRON consists of a directed graph $G = (V, E)$, with two specified starting vertices $s_1, s_2 \in V$. Players must alternately choose a vertex, that has not been chosen before, and that must have an incoming edge with the last chosen vertex *by that player* as its other endpoint. The first vertex, chosen by player 1 must be s_1, and the first vertex, chosen by player 2 must be s_2. The first player that is unable to move loses the game.

In other words, two disjoint simple paths are formed, one by player 1 and one by player 2. A variant where the starting vertices s_1 and s_2 are not specified (i.e.,

players are free to choose their first vertex), is also considered. This game TRON can be seen as a generalization of a popular video game, based upon a Walt Disney movie of the same name. In this video game, the game is basically played "in real time" on a large complete grid graph.

In this paper, we will also consider variants, where the games are played on undirected graphs.

There is a close connection between VERTEX and EDGE GENERALIZED GEOGRAPHY and games with rules, forbidding positions on moves to appear more than once. A game can be modeled by a directed graph, where the vertices correspond to positions in the game. Edges correspond to possible moves from a position. (For most games, this graph has a very large size.) If it is forbidden to move to a position that has appeared already earlier in the game, then this corresponds to the condition that the players alternately choose a vertex on a simple path; a rule that forbids the same move from the same position corresponds to an elementary path. (Compare [34]).

3 PSPACE-completeness results.

In this section we give a number of new PSPACE-completeness results. We establish PSPACE-completeness for the following games/problems:

- VERTEX GENERALIZED GEOGRAPHY without specified starting vertex
- HAMILTONIAN PATH CONSTRUCTION GAME with and without specified starting vertex
- HAMILTONIAN CIRCUIT CONSTRUCTION GAME with and without specified starting vertex
- TRON with and without starting vertices
- SIMPLE PATH CONSTRUCTION GAME with and without specified starting vertices for directed and for undirected graphs
- ELEMENTARY PATH CONSTRUCTION GAME with and without specified starting vertex for directed and for undirected graphs

In each of our proofs, either we use a transformation from QUANTIFIED 3-SATISFIABILITY similar to the proof for (VERTEX or EDGE) GENERALIZED GEOGRAPHY in [29, 36], or a transformation from a closely related game.

Theorem 3.1
VERTEX GENERALIZED GEOGRAPHY without specified starting vertex is PSPACE-complete.

Proof.
Clearly the problem is in PSPACE. To prove PSPACE-hardness, we use a transformation from the standard VERTEX GENERALIZED GEOGRAPHY problem (i.e., with specified starting vertex).

Let an instance $G = (V, E)$, $s \in V$ of VERTEX GENERALIZED GEOGRAPHY be given. We may suppose that the indegree of s is 0 (it is never possible to traverse an edge to the starting vertex, so these edges may as well be deleted from G).

Now let $G' = (V', E')$ be defined as follows: (see Figure 3.1).

$$
\begin{array}{lll}
V' = V & \cup & \{0, 1_a, 1_b\} \cup \\
 & & \{v^+ \mid v \in V \text{ and } v \neq s\} \\
E' = E & \cup & \{(v, v^+) \mid v \in V \text{ and } v \neq s\} \\
 & \cup & \{(v^+, v) \mid v \in V \text{ and } v \neq s\} \\
 & \cup & \{(v^+, 1_a) \mid v \in V \text{ and } v \neq s\} \\
 & \cup & \{(v^+, 1_b) \mid v \in V \text{ and } v \neq s\} \\
 & \cup & \{(1_a, s), (1_b, s), (0, 1_a), (0, 1_b), (1_a, 0), (1_b, 0)\}.
\end{array}
$$

As the construction of G' can be carried out in logarithmic work space, the theorem follows with help of the following claim. □

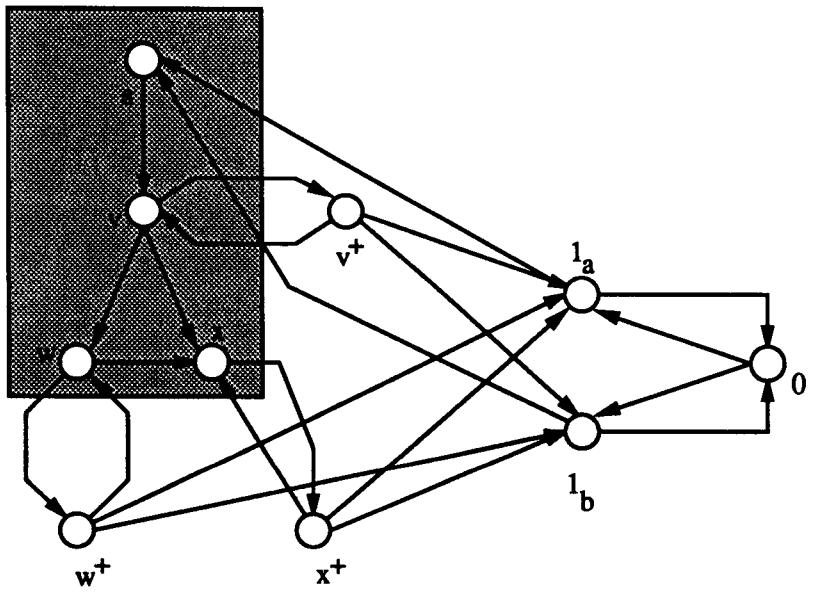

Figure 3.1

Claim 3.2
There is a winning strategy for player 1 on G for VERTEX GENERALIZED GEOGRAPHY with starting vertex s, if and only if there is a winning strategy for player

1 on G' for VERTEX GENERALIZED GEOGRAPHY without specified starting vertex.

Proof.
Suppose player 1 has a winning strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. Then he can use the following strategy for VERTEX GENERALIZED GEOGRAPHY without specified starting vertex on G': start in 0. Player 2 will move to 1_a or 1_b. Move to s. Now, as long as player 2 moves to vertices in V, also move to vertices in V, using the original strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. After a number of moves, player 2 will be unable to move to a vertex in V. So player 2 will move eventually to a vertex v^+. Now player 1 moves to the unused vertex in $\{1_a, 1_b\}$ and wins the game.

Suppose player 2 has a winning strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. Player 2 now also has a winning strategy for VERTEX GENERALIZED GEOGRAPHY without specified starting vertex on G'. We consider a number of cases.

CASE 1. If player 1 starts at 0, then player 2 wins, using a strategy, similar to the argument above.

CASE 2. Suppose player 1 starts at 1_a or 1_b. W.l.o.g. suppose player 1 starts at 1_a. Player 2 moves to s. Player 1 must move to a $v \in V$, $v \neq s$. Player 2 moves to v^+. Player 1 must move to 1_b. Player 2 moves to 0 and wins.

CASE 3. Suppose player 1 starts at $v \in V$, $v \neq s$. Then player 2 moves to v^+. Player 1 must move to 1_a or 1_b. Player 2 moves to 0. Player 1 must move to the unused vertex in $\{1_a, 1_b\}$. Player 2 moves to s. Player 1 must move to a vertex $v \in V$, $v \neq s$. Player 2 moves to v^+ and wins the game.

CASE 4. Player 1 starts at a vertex v^+, $v \in V$. Player 2 moves to v. Player 1 must move to a vertex $w \in V$. $w \neq s$, because indegree(s)=0. Player 2 moves to w^+. Player 1 must move to 1_a or 1_b. Player 2 moves to 0. Player 1 must move to the free vertex in $\{1_a, 1_b\}$. Player 2 moves to s. Player 1 must move to a vertex $x \in V$, $x \neq v$, $x \neq w$. (If x does not exist, player 1 loses directly). Now player 2 moves to x^+ and wins the game.

CASE 5. Player 1 starts at vertex s. Player 2 now follows the strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G, as long as player 1 moves to vertices $v \in V$. Eventually, player 1 must move to a vertex v^+. Now player 2 moves to 1_a, player 1 to 0, and player 2 wins by moving to 1_b. □

Corollary 3.3
VERTEX GENERALIZED GEOGRAPHY without specified starting vertex is PSPACE-complete for graphs with thickness ≤ 2.

Proof.
VERTEX GENERALIZED GEOGRAPHY with specified starting vertex is

PSPACE-complete for planar graphs [29]. If the construction in the proof of Theorem 3.1 is applied to a planar graph, one obtains a graph with thickness ≤ 2. □

Theorem 3.4
HAMILTONIAN PATH CONSTRUCTION GAME with specified starting vertex is PSPACE-complete.

Proof.
Clearly the problem is solvable in polynomial space. To prove PSPACE-hardness, we use a transformation from QUANTIFIED 3-SATISFIABILITY. Let an instance of this problem be given. W.l.o.g., we may suppose it is of the form

$$F = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n F_0,$$

with F_0 a boolean expression in conjunctive normal form. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in F_0. W.l.o.g. we may suppose that $m \geq 4$.

Let $G = (V, E)$ be the directed graph, defined by

$$V = \{s, t\} \cup \{x_i \mid 1 \leq i \leq n\} \cup \{\overline{x_i} \mid 1 \leq i \leq n\} \cup \{r_1, r_2\} \cup \{c_i \mid 1 \leq i \leq m\}$$

$$\begin{aligned} E = {} & \{(s, x_1), (s, \overline{x_1}), (t, x_1), (t, \overline{x_1}), (x_n, r_1), (x_n, r_2)\} \cup \{(c_i, t) \mid 1 \leq i \leq m\} \cup \\ & \{(r_i, c_j) \mid i = 1, 2,\ 1 \leq j \leq m\} \cup \\ & \{(c_i, c_j) \mid i \neq j, 1 \leq i, j \leq m\} \cup \\ & \{(x_i, x_{i+1}) \mid 1 \leq i < m\} \cup \{(x_i, \overline{x_{i+1}}) \mid 1 \leq i < m\} \cup \\ & \{(\overline{x_i}, x_{i+1}) \mid 1 \leq i < m\} \cup \{(\overline{x_i}, \overline{x_{i+1}}) \mid 1 \leq i < m\} \cup \\ & \{(c, l) \mid l \text{ is a literal, appearing in clause } c\} \end{aligned}$$

(See Figure 3.2).

Claim 3.5
There is a winning strategy for player 1 in the HAMILTONIAN CIRCUIT CONSTRUCTION GAME on G with starting vertex s, if and only if F is false.

Proof.
Suppose F is false. We give a winning strategy for player 1. Player 2 will choose x_1 or $\overline{x}_1$, then player 1 chooses x_2 or $\overline{x}_2$, etc. Call vertices which are chosen (before the negation is chosen) false. As F is false, player 1 can choose the x_i's (i even) in such a way, that whatever strategy player 2 will use, there will be at least one clause with only false variables, say clause c_{i_0}. Now, after from each pair $x_i, \overline{x_i}$ one has been visited, player 2 will visit r_1 or r_2. Then player 1 goes to c_{i_0}. Player 2 must go either to t, or to another c_i. In the latter case, player 1 moves to t. Now from t all vertices x_i, $\overline{x_i}$ that have not yet been visited will be visited, then the vertex in $\{r_1, r_2\}$ that has not yet been visited, and then all unvisited vertices c_i. So player 1 wins the game, as all vertices will eventually be visited.

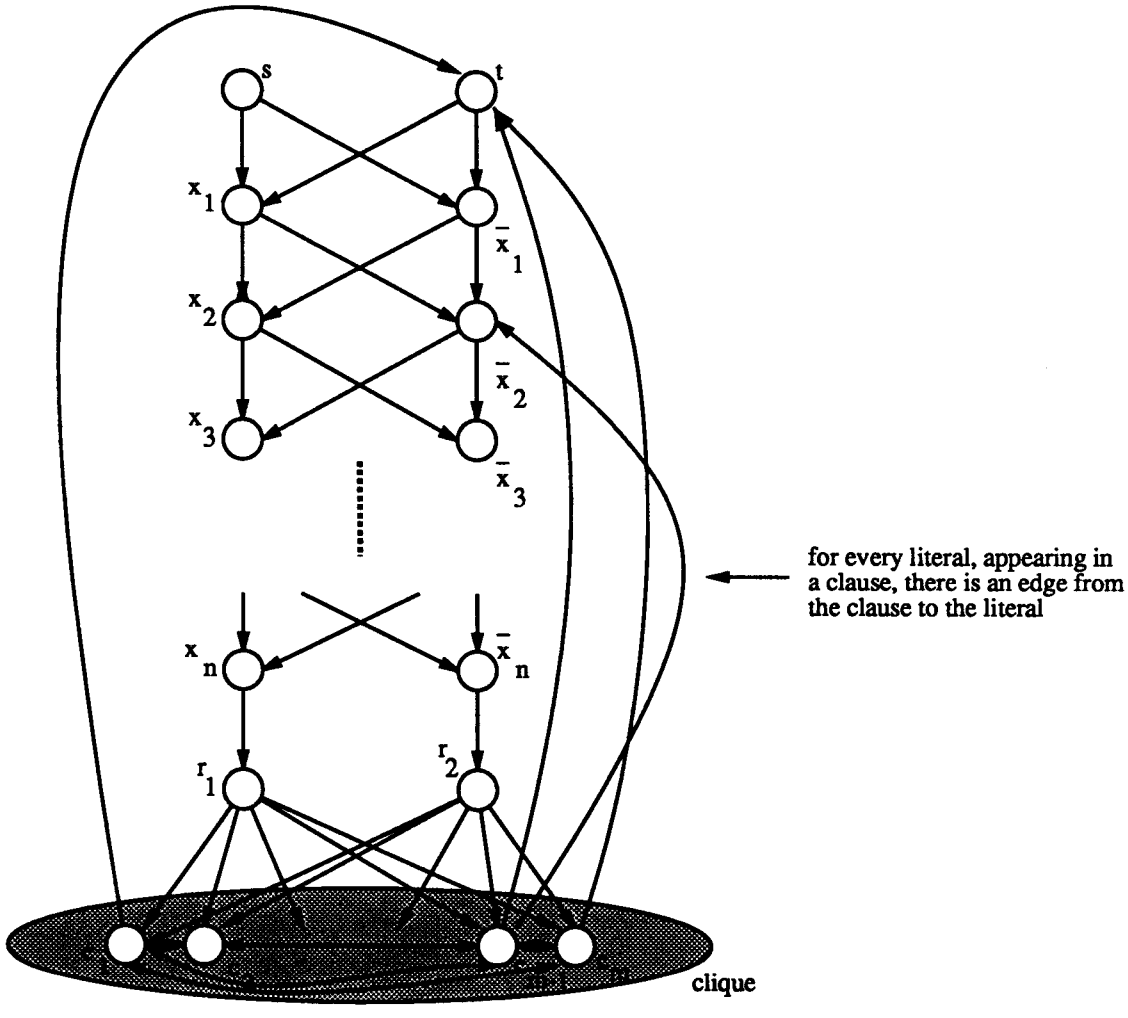
s
t
x_1
$\bar{x}_1$
x_2
$\bar{x}_2$
x_3
$\bar{x}_3$
x_n
$\bar{x}_n$
r_1
r_2
for every literal, appearing in a clause, there is an edge from the clause to the literal
clique

Figure 3.2

Suppose F is true. Now player 2 can force that at the first moment a vertex c_{i_0} is visited, each clause contains at least one literal, corresponding to a vertex that is not already visited. Player 1 has moved to c_{i_0}, and then player 2 moves to such a vertex x_i or $\overline{x}_i$. Now all yet unvisited vertices x_j, $\overline{x}_j$ with $j > i$ are visited, then the remaining vertex in $\{r_1, r_2\}$, and then a player moves to a vertex c_i. If player 2 now may move, he moves to t. If player 1 now may move, and moves to another c_i, then player 2 moves from this c_i to t. Now the game will stop before all vertices c_i have been visited. (At most 3 c_i's are visited and $m \geq 4$.) So player 2 wins the game. □

As the construction of G can be done in logarithmic working space, the theorem follows. □

Theorem 3.6
HAMILTONIAN CIRCUIT CONSTRUCTION GAME with specified starting vertex is PSPACE-complete.

Proof.
Use the construction of Theorem 3.4, but add an edge from each c_i to s. □

Theorem 3.7
HAMILTONIAN PATH CONSTRUCTION GAME without specified starting vertex is PSPACE-complete.

Proof.
Look at the proof of Theorem 3.4. Note that player 1 must start in s, because indegree(s) = 0. □

Theorem 3.8
HAMILTONIAN CIRCUIT CONSTRUCTION GAME without specified starting vertex is PSPACE-complete.

Proof.
Use the construction of Theorem 3.6 (i.e., with an edge from each c_i to s). If player 1 starts at s or t, then the game is as with specified starting vertex s. If player 1 does not start at s or t, then player 2 can win if $m \geq 5$: he always moves to s or t when possible. At least one vertex c_i now will be unvisited at the end of the game. □

Corollary 3.9
SIMPLE PATH CONSTRUCTION GAME with or without specified starting vertex is PSPACE-complete.

Theorem 3.10
SIMPLE PATH CONSTRUCTION GAME with specified starting vertex is PSPACE-complete for undirected graphs.

Proof.
Clearly, the problem is in PSPACE. We use again a transformation of QUANTIFIED 3-SATISFIABILITY to prove PSPACE-hardness. Let an instance of Q-3-SAT:

$$F = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n F_0$$

be given; F_0 is a boolean formula in conjunctive normal form. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in F_0. Assume that $\forall x_i : \exists c \in C : x_i \in c$; $\exists c \in C \overline{x}_i \in C$. Let $G = (V, E)$ be the following graph:

$$\begin{aligned}
V = \; & \{v_i \mid 1 \leq i \leq n\} \cup \\
& \{x_i \mid 1 \leq i \leq n\} \cup \{\overline{x}_i \mid 1 \leq i \leq n\} \cup \\
& \{y_i \mid 1 \leq i \leq n\} \cup \{\overline{y}_i \mid 1 \leq i \leq n\} \cup \\
& \{w_i \mid 1 \leq i \leq n\} \cup \{z_i \mid 1 \leq i \leq n\} \cup \\
& \{c_i \mid 1 \leq i \leq m\} \cup \{r\} \cup \\
& \{d_{ijk} \mid 1 \leq i \leq m,\ 1 \leq j \leq 3,\ 1 \leq k \leq K\} \\
E = \; & \{(v_i, x_i), (v_i, \overline{x}_i), (x_i, y_i), (\overline{x}_i, \overline{y}_i), (y_i, w_i), (\overline{y}_i, w_i) \mid 1 \leq i \leq n,\ i \text{ odd}\} \cup \\
& \{(v_i, y_i), (v_i, \overline{y}_i), (y_i, x_i), (\overline{y}_i, \overline{x}_i), (x_i, w_i), (y_i, w_i) \mid 1 \leq i \leq n,\ i \text{ even}\} \cup \\
& \{(w_i, z_i) \mid 1 \leq i \leq n\} \cup \\
& \{(z_i, v_{i+1}) \mid 1 \leq i < n\} \cup \\
& \{(z_n, c_i) \mid 1 \leq i \leq m\} \cup \\
& \{(c_i, d_{ij1}) \mid 1 \leq i \leq m,\ 1 \leq j \leq 3\} \cup \\
& \{(d_{ijk}, d_{ij(k+1)}) \mid 1 \leq i \leq m,\ 1 \leq j \leq 3,\ 1 \leq k \leq K\} \cup \\
& \{(d_{ij1}, l) \mid l \text{ is the } j\text{'th literal, appearing in clause } c_i\}
\end{aligned}$$

where $K = 5n + 8$. (See Figure 3.3). The starting vertex is v_1.

Claim 3.11
F is true, if and only if player 1 can force a path with length $\geq K$.

Proof.
Note that the resulting path will have length $\geq K$, if and only if a "long branch" $d_{ij1} d_{ij2} \ldots d_{ijK}$ is used. (Every other path in G has length $< K$.) Next note that players that must move from a vertex w_i must move to z_i or lose the game: if player 1 must move from w_i, then i is even. If he moves to x_i or $\overline{x}_i$, then player 2 moves to y_2 or $\overline{y}_2$, and the game ends with a path, shorter than K. If player 2 must move from w_i then i is odd. If he moves to y_i or $\overline{y}_i$, then player 1 moves to x_i or $\overline{x}_i$; player 2 then must move to a vertex d_{ij1}, and then player 1 moves to d_{ij2} and forces a path with length $\geq K$.

The proof proceeds with arguments, which are similar to arguments used before. Let a *used* x_i or $\overline{x}_i$ correspond with <u>true</u>. Player 1 tries to have in each clause a

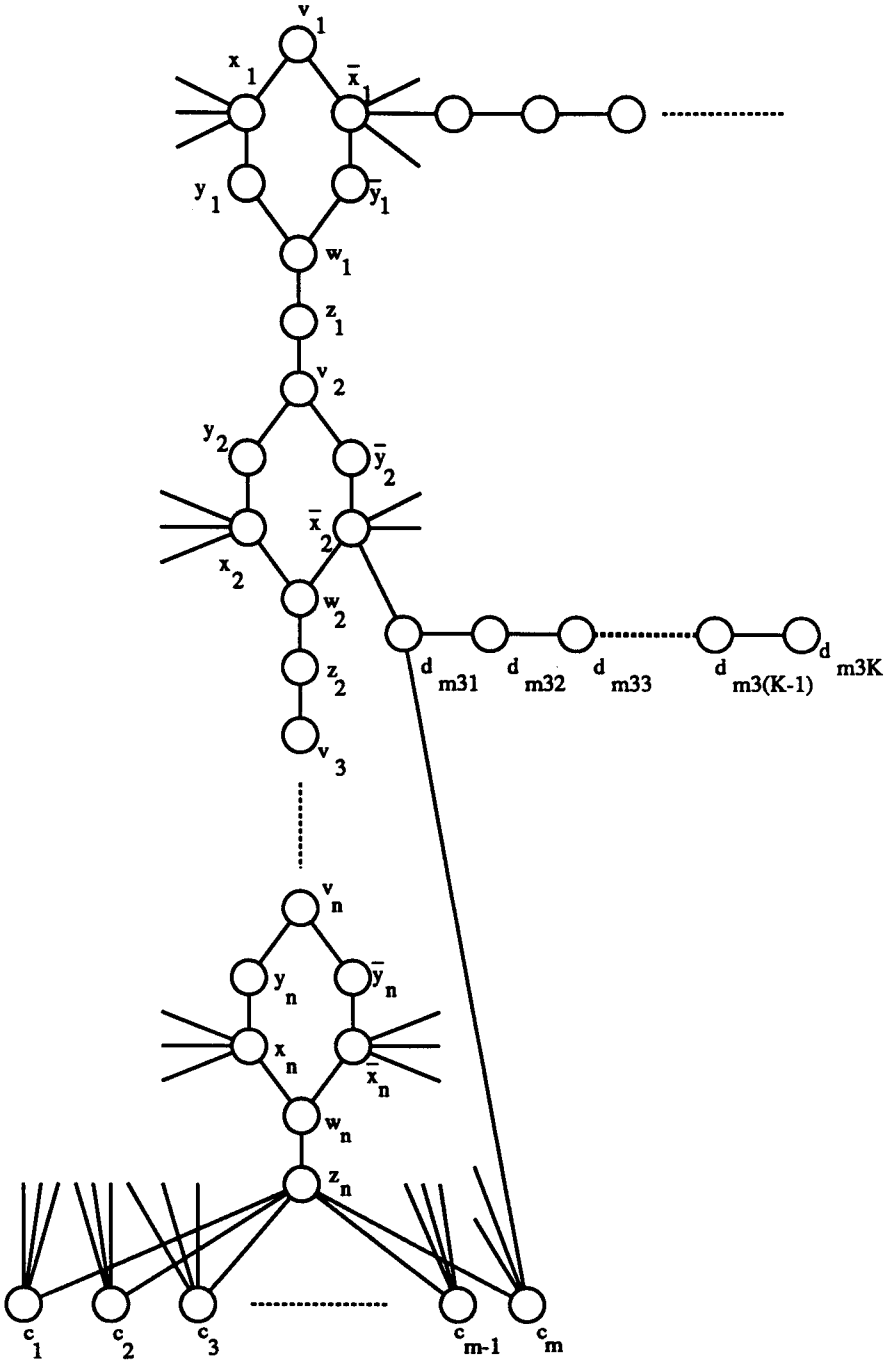
v 1
x 1
x̄ 1
y 1
ȳ 1
w 1
z 1
v 2
y 2
ȳ 2
x̄ 2
x 2
w 2
z 2
v 3
d m31
d m32
d m33
d m3(K-1)
d m3K
v n
y n
ȳ n
x n
x̄ n
w n
z n
c 1
c 2
c 3
c m-1
c m

Figure 3.3

literal l, with the corresponding vertex l is visited when the path reaches z_n. He succeeds if and only if the formula is true. Player 2 must move from z_n. He will move, if possible to a unsatisfied clause, i.e., each literal in the clause is unvisited. Player 1 will move to a vertex d_{ij1}. If the corresponding literal in the clause is true (visited), then player 2 must move to d_{ij2}, and the resulting path has length $\geq K$. Otherwise, player 2 can move to that literal-vertex, and the resulting path has length $< K$. □

We have obtained a log-space transformation from QUANTIFIED 3-SATISFIABILITY to SIMPLE PATH CONSTRUCTION GAME for undirected graphs. Hence the latter is PSPACE-complete. □

Theorem 3.12
SIMPLE PATH CONSTRUCTION GAME without specified starting vertex is PSPACE-complete for undirected graphs.

Proof.
To prove PSPACE-hardness, we use a transformation from the case with specified starting vertex. Let an instance $G = (V, E)$, $s \in V$, $K \in \mathbb{N}^+$ of the latter problem be given. Let $G' = (V', E')$ be defined as follows (see Figure 3.4).

$$\begin{aligned} V' = \; & V \cup \{r_i \mid 1 \leq i \leq 2 \cdot |V| + 2\} \\ & \cup \{q_i \mid 1 \leq i \leq K + 2\} \cup \{y\} \\ E' = \; & E \cup \{(r_i, r_{i+1}) \mid 1 \leq i \leq 2|V|\} \\ & \cup \{(q_i, q_{i+1}) \mid 1 \leq i \leq K + 1\} \\ & \cup \{(r_{2|V|+2}, s), (r_{2|V|+1}, q_1), (r_{2|V|}, y)\} \end{aligned}$$

Claim 3.13
Player 1 can force a path with length $\geq K + 2|V| + 2$ on G' with no specified starting vertex, if and only if player 1 can force a path with length $\geq K$ on G with starting vertex s.

Proof.
"$\Leftarrow$": Player 1 starts at r_1. When player 2 must move eventually from $r_{2|V|+1}$, he can go to q_1 (in which case the resulting path has the required length), or go to $r_{2|V|+2}$. In the latter case, player 1 now can use the strategy for the game on G with starting vertex s.
"$\Rightarrow$". We consider several cases for the start of player 1 on G'.
CASE 1. Player 1 starts at a vertex q_i. Then he will lose: either the path ends at q_{K+2} (with length $\leq K + 2$), or a player moves to $r_{2|V|+1}$. If player 1 moves from $r_{2|V|+1}$ to $r_{2|V|}$, the player 2 moves to y and wins. If any player moves from $r_{2|V|+1}$

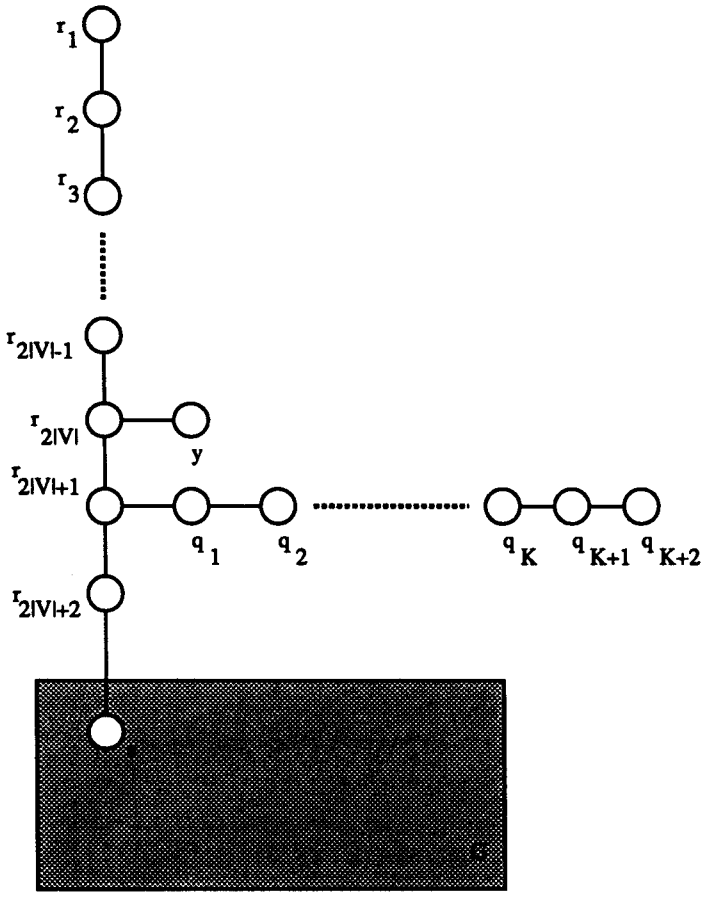

Figure 3.4

to $r_{2|V|+2}$, the resulting path will have length $\leq K + 4 + |V|$.
CASE 2. Player 1 starts at a vertex r_i, $i \neq 1$. If i is odd, player 2 will eventually move to q_1, if i is even, then player 2 will eventually move to y. In both cases, the resulting path has length $\leq K + 2|V|$.
CASE 3. Player 1 starts at y. No matter what strategy is used by either player, the resulting path will have length $\leq 2|V| + 1$.
CASE 4. Player 1 starts at a vertex $v \in V$. After at most $|V|$ moves, a player will move to $r_{2|V|+2}$, or the resulting path has length $\leq |V|$. Player 2 uses the strategy to move to q_1 or y if possible. The resulting path will have length $\leq |V| + 2 + K + 2$.
CASE 5. Player 1 starts at q_1. If player 2 moves from $r_{2|V|+1}$ to q_1, then player 1 succeeds. If player 2 moves from $r_{2|V|+1}$ to $r_{2|V|+2}$, then player 1 succeeds exactly if he can force a path with length K in G with starting vertex s. □

Again this transformation can be done in logarithmic working space. □

Theorem 3.14
ELEMENTARY PATH CONSTRUCTION GAME (with specified starting vertex) is PSPACE-complete.

Proof.
The proof is similar to the proof of Theorem 3.10. A variable x_i with i odd is replaced by the construction of Figure 3.5.a., and a variable with i even will be replaced by the construction of Figure 3.5.b. In the "first pass", players will move from a_i to d_i and from d_i to a_{i+1}. Player 2 will never take a "side-branch", because then he loses the game. Player 1 will never move from d_i to x_i or $\overline{x}_i$, because then player 2 will move, such that the game stops after 8 or 9 moves in a_i. An unvisited x_i corresponds to true. Player 1 can move to a vertex on a branch before an unvisited x_i, if and only if the formula is true. If x_i is visited, then player 2 moves from the vertex on the branch to x_i, and the game stops. Otherwise, player 1 can move such that he can go from c_i to the branch with length K, attached to c_i. We omit the details. □

Theorem 3.15
ELEMENTARY PATH CONSTRUCTION GAME without specified starting vertex is PSPACE-complete.

Proof.
This follows with a construction, similar to the construction of the proof of Theorem 3.12. □

Theorem 3.16
TRON with specified starting vertices is PSPACE-complete.

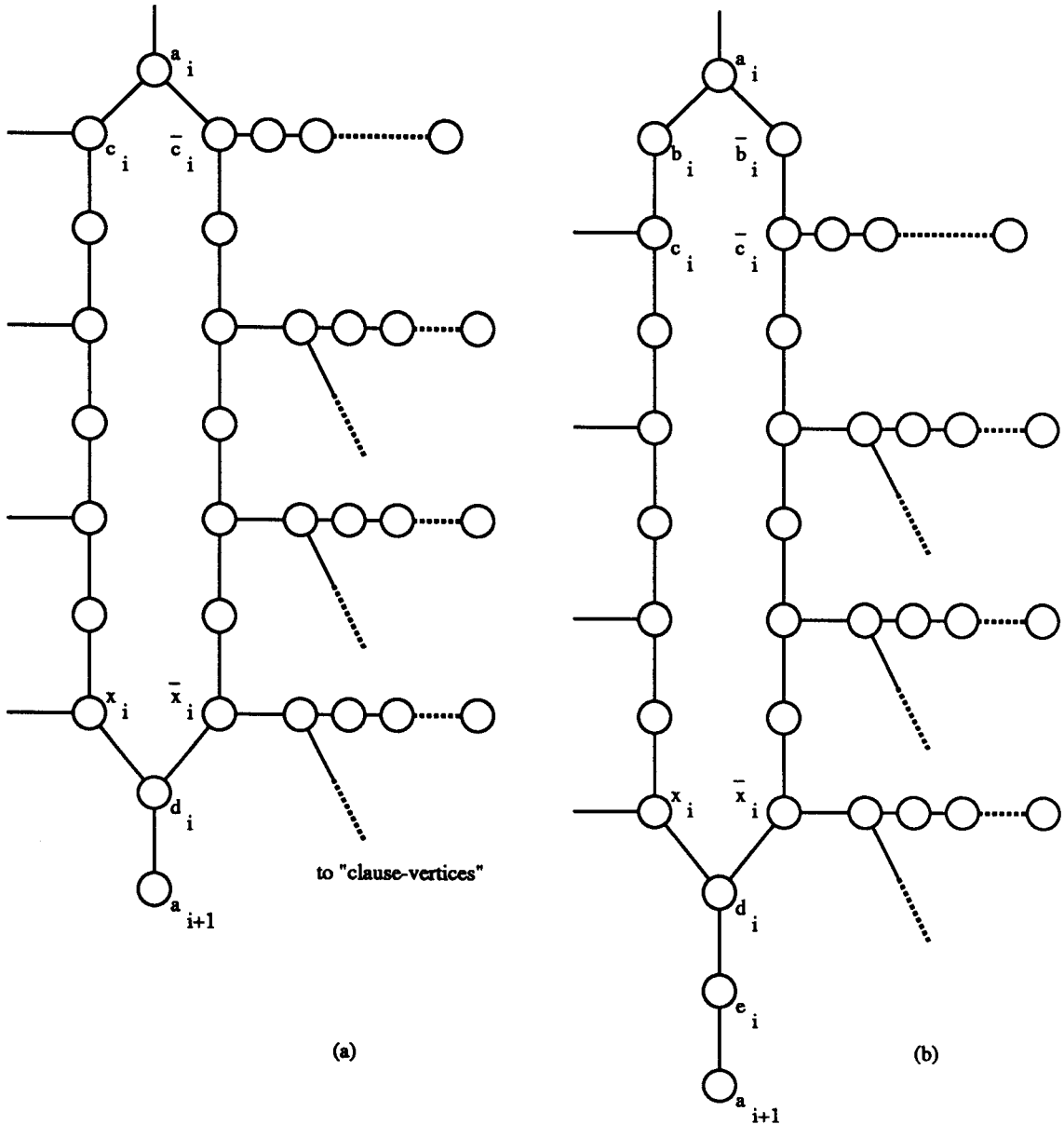
a i
c i
c̄ i
x i
x̄ i
d i
to "clause-vertices"
a i+1
(a)
a i
b i
b̄ i
c i
c̄ i
x i
x̄ i
d i
e i
a i+1
(b)

Figure 3.5

Proof.
Clearly TRON is solvable in polynomial space. (Use backtracking and note that there are at most n moves in any game.)

To prove PSPACE-hardness, we use a transformation from QUANTIFIED 3-SATISFIABILITY. Let an instance of Q-3-SAT be given. We may assume that this instance is of the form

$$F = \exists X_1 \forall X_2 \exists X_3 \forall X_4 \ldots \exists X_{n-1} \forall X_n F_0,$$

where F_0 is in conjunctive normal form with 3 literals per clause. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in F_0. Now define $G = (V, E)$ by:

$$\begin{aligned}
V = &\{\ s_1, s_2, r, t\} \cup \\
&\{\ x_i \mid 1 \leq i \leq n\} \cup \{\overline{x}_i \mid 1 \leq i \leq n\} \cup \\
&\{\ y_i \mid 1 \leq i \leq n\} \cup \\
&\{\ z_i \mid 1 \leq i \leq m\} \cup \\
&\{\ c_i \mid 1 \leq i \leq m\} \\
E = &\{\ (s_1, x_1), (s_1, \overline{x}_1), (s_2, y_1), (y_n, z_1), (x_n, r), (\overline{x}_n, r)\} \cup \\
&\{\ (x_i, y_{i+1}) \mid 1 \leq i < n\} \cup \{(\overline{x}_i, y_{i+1}) \mid 1 \leq i < n\} \cup \\
&\{\ (y_i, x_{i+1}) \mid 1 \leq i < n\} \cup \{(y_i, \overline{x}_{i+1}) \mid 1 \leq i < n\} \cup \\
&\{\ (z_i, z_{i+1}) \mid 1 \leq i < m\} \cup \{(r, c_i) \mid 1 \leq i < m\} \cup \\
&\{\ (c_i, c_j) \mid i \neq j, 1 \leq i, j \leq m\} \cup \\
&\{\ (l, c_i) \mid l \text{ of form } x_j \text{ or } \overline{x}_j,\ 1 \leq i \leq m,\ l \text{ appears in clause } c_i\}
\end{aligned}$$

In Figure 3.6 we give a graphical representation of G.

Claim 3.17
F is true, if and only if there is a winning strategy for player 1, on G with starting vertices s_1, s_2.

Proof.
Suppose that F is true. Note that first player 1 decides whether to take x_1 or $\overline{x}_1$, then player 2 decides whether to take x_2 or $\overline{x}_2$, etc. As F is true, player 1 can move in such a way, that when player 1 has moved to z_1 and player 2 has moved to r, then for each clause $c \in C$, there is at least one vertex, corresponding to a literal $l \in c$, that is not yet visited. (Moving to a variable corresponds to making that variable false.) Now player 1 will move to z_2. Then player 2 moves to a c_i. If player 2 moves to a vertex x_i, $\overline{x}_i$ or to t, before player 1 moves from z_m to a vertex c_i, then he loses. When player 1 moves to a vertex c_{i_0}, then each vertex c_i has been visited ($m-1$ of these by player 2, and 1 by player 1). Now player 2 must move to a x_i, $\overline{x}_i$ or to t. If player 2 moves to a vertex $\neq t$, then player 1 moves to t and wins. If player 2 moves to t, then note that there is at least one literal $l \in c_{i_0}$ with vertex l not yet visited. So player 1 can move to that vertex and wins.

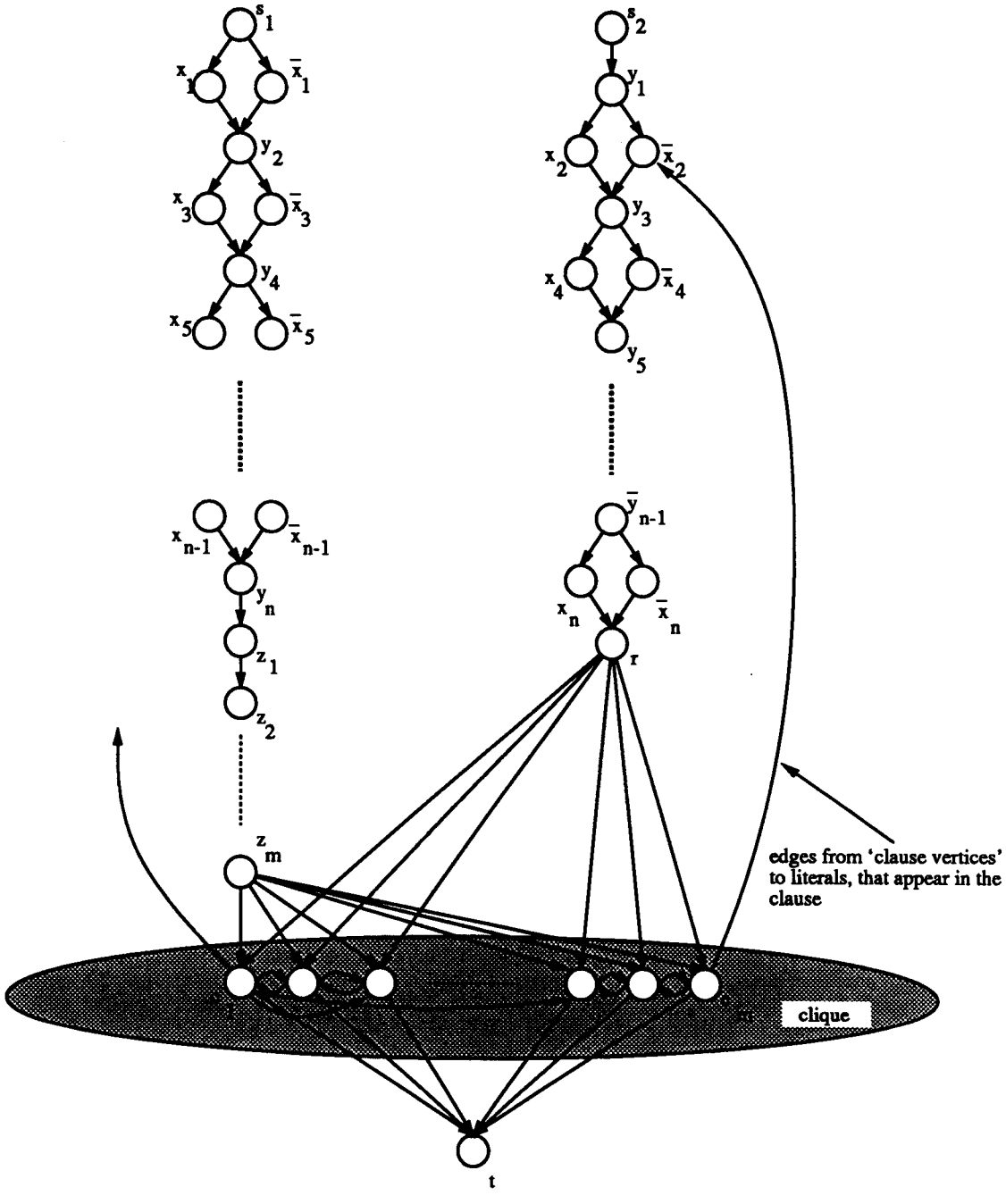
s1
x1
x̄1
y2
x3
x̄3
y4
x5
x̄5
s2
y1
x2
x̄2
y3
x4
x̄4
y5
xn-1
x̄n-1
yn
z1
z2
zm
ȳn-1
xn
x̄n
r
edges from 'clause vertices' to literals, that appear in the clause
clique
t

Figure 3.6

Now suppose that F is false. Player 2 now can make sure that when player 1 is on z_1 and player 2 is on r, then there is at least one clause $c \in C$ with all vertices, corresponding to literals in c, have been visited. Let this clause be c_{i_0}. Then, while player 1 is visiting all vertices on the path $z_2 \rightarrow z_3 \rightarrow \ldots \rightarrow z_m$, player 2 visits all c_i with $i \neq i_0$. Then player 1 must move to c_{i_0}. Player 2 moves to t. As all vertices, corresponding to literals in c_{i_0} have been visited, player 1 cannot move and loses the game. □

As the construction of G can be done with logarithmic working space, it follows that TRON with specified starting vertices is PSPACE-complete. □

Theorem 3.18
TRON without specified starting vertices is PSPACE-complete.

Proof.
We modify the construction of the proof of Theorem 3.16 a little. We add vertices $\{s_1^*, s_2^*, t_1, \ldots, t_{5+2n+2m}\}$ and edges (s_1^*, s_1), (s_2^*, s_2), (s_1^*, s_2^*), (t, s_1), (s_1^*, t_1), (t_i, t_{i+1}) $(1 \leq i \leq 4 + 2n + 2m)$. (See Figure 3.7.)

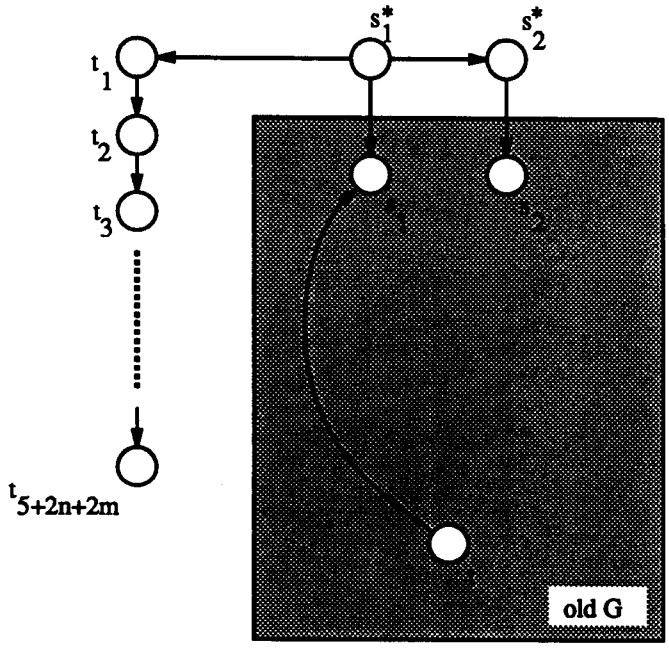

Figure 3.7

Note that the longest simple path in the resulting graph has length $6 + 2n + 2m$: start in s_1^*, go to s_2^*, s_2, via y_i's, x_0's and $\overline{x}_i$'s to r, visit $m - 1$ c_i's, go to t, s_1, via

y_i's, x_i's and $\overline{x}_i$'s to z_1, go to z_m, to the last c_i, and to a literal $\in c_i$ that is not visited (one can always do this).

Now: if player 1 does not start at s_1^*, he loses the game: player 2 starts at s_1^*. If player 1 started at a vertex t_i, then player 2 will walk the simple path with length $6 + 2n + 2m$ and wins the game. If player 1 started at a vertex not in $\{t_1, \ldots, t_{5+2n+2m}\}$, then player 2 will visit all t_i's. Player 1 can make at most $5 + 2n + 2m$ moves, and hence player 2 wins the game.

Suppose player 1 starts at s_1^*. Then player 2 must start at s_2^*. Suppose he does not. With a strategy, similar to the previous argument, player 1 can now win the game.

Now suppose player 1 starts at s_1^*, and player 2 starts at s_2^*. If player 1 moves from s_1^* to t_1, he loses: player 2 can walk a simple path with length $5 + 2n + 2m$, and hence can make the last move in the game.

So player 1 must start at s_1^*, then player 2 starts at s_2^*, then player 1 moves to s_1, then player 2 moves to s_2. Now the situation is identical to the game with specified starting vertices s_1, s_2. □

We end this section with a small comment on the standard VERTEX and EDGE GENERALIZED GEOGRAPHY GAMES. Clearly, when we restrict ourselves to *acyclic graphs*, then the problems are easy to resolve in $\mathcal{O}(n + e)$ time. However, the proof in [29, 36] for the PSPACE-completeness of VERTEX or EDGE GENERALIZED GEOGRAPHY can easily be modified, such that we have PSPACE-completeness for the problems on graphs, obtained by *adding one edge to an acyclic graph.*

4 Polynomial time algorithms for path-forming games on special classes of graphs.

In this section we give polynomial time algorithms for several of the considered games on special classes of graphs. In Section 4.1 we give linear algorithms for some of the games on graphs with bounded treewidth, based upon an intricate characterization of subgraphs, and dynamic programming. In Section 4.2 we show how graph rewriting can be employed to solve some problems on cacti.

4.1 Linear time algorithms for some games on graphs with bounded treewidth.

In this section we show how VERTEX GENERALIZED GEOGRAPHY, HAMILTONIAN PATH CONSTRUCTION GAME and HAMILTONIAN CIRCUIT CONSTRUCTION GAME can be solved in linear time on graphs with a fixed upper-

bound k on the treewidth. We first consider VERTEX GENERALIZED GEOGRAPHY.

Let in the remainder of this section k be a constant. We will assume that input graphs $G = (V, E)$ are given with a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G with treewidth $\leq k$. If not, then such a tree-decomposition can be found (if it exists) in $\mathcal{O}(n^{k+2})$ time with dynamic programming [3] or in $\mathcal{O}(n^2)$ time with graph minor theory and self-reduction [11]. For $k = 1, 2, 3$, the tree-decomposition can be found in linear time [5, 30].

It is not difficult to see that one may assume that the tree T in the tree-decomposition is binary (e.g. use the transformation used in [10]). In the remainder we assume that we have a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of $G = (V, E)$ with treewidth $\leq k$, and T a binary tree. We also suppose that there exists an $i_0 \in I$ with $X_{i_0} = \{s\}$. (Take an arbitrary $i \in I$ with $s \in X_i$. Add a branch (i, i_0) to T with $X_{i_0} = \{s\}$. A correct tree-decomposition results. Now apply the technique to make T a binary tree.) i_0 is taken as root of T.

We now give an inductive definition of charac(X, W), for $X, W \subseteq V$. The resulting algorithm will compute for all $i \in I$: charac(X_i, Y_i), where $Y_i = \{v \in X_j \mid j$ a descendant of $i\} - X_i$. From charac(X_{i_0}, Y_{i_0}) = charac$(\{s\}, V - \{s\})$ the answer to the problem can be determined quickly. charac(X, W) is defined with induction to $|X|$.

For sets X, define $C(X)$ to be the set of all possible values of charac(X, W) over all graphs $G = (V, E)$, $X, W \subseteq V$, $X \cap W = \emptyset$.

If $X = \emptyset$, then charac(X, W) is the empty string, i.e., for $X = \emptyset : C(X) = \{\varepsilon\}$, ε the empty string.

If $|X| = 1$, then charac(X, W) is a boolean $\in$ {true,false}, that denotes whether there is a winning strategy for player 2 for VERTEX GENERALIZED GEOGRAPHY, played on $G[X \cup W]$, the subgraph of G induced by $X \cup W$, with starting vertex the unique vertex $x \in X$.

Now suppose $|X| \geq 2$. We first must introduce some other notions. Let $x \in X$. Consider the following type of variant of VERTEX GENERALIZED GEOGRAPHY: the game ends when a player moves to a vertex $y \in X$, or when a player cannot make a move. In the former case, let $W' \subseteq W$ be the set of vertices in W not yet visited, and consider charac$(X - \{x, y\}, W') \in C(X - \{x, y\})$. Let $P(X, W, x, y)$ be the set of all pairs (p, c) with $p \in \{1, 2\}$ denoting a player, and $c \in C(X - \{x, y\})$, such that there is a possible play by players 1 and 2 in the above type of game, starting at x, where player p moves to y, and c = charac$(X - \{x, y\}, W')$ with W' the set of vertices in W that are not visited in the game. Let $P(X, W, x)$ be the set of all triples (p, c, y) with $y \in X - \{x\}$ and $(p, c) \in P(X, W, x, y)$. For each $R \subseteq P(X, W, x)$ we now consider the game $VCC(X, W, x, R)$. This is the variant of VERTEX GENERALIZED GEOGRAPHY described above, with the following properties: player 1 starts with moving from x to a vertex in $W \cup X$. The game ends when a player cannot make a move from a vertex in W — then this player loses the game — or when a player j moves to a vertex $y \in X$, with W' is the set of

vertices in W that are not visited. In this case, player 1 wins the game, if and only if $(j, \text{charac}(X - \{x, y\}, W'), y) \in R$.

We can now describe charac(X, W) for $|X| \leq 2$. charac(X, W) is a pair (f_1, f_2), where

- f_1 maps each pair (x, R) with $R \subseteq \bigcup_{y \in X - \{x\}} \{(p, c, y) \mid p \in \{1, 2\},\ c \in C(X - \{x, y\})\}$ to a boolean, that is true, if and only if $R \subseteq P(X, w, x)$ and there is a winning strategy for player 1 in the game $VCC(X, W, x, R)$.
- f_2 maps each $x \in X$ to charac$(X - \{x\})$.

(The case with $|X| = 1$ can be seen as a special case of the above definition. Here $P(X, W, x) = \emptyset$, so f_1 must only state whether there is a winning strategy for player 1 in $VCC(\{x\}, W, x, \emptyset)$, which equals charac$(\{x\}, W)$. f_2 maps x to the empty string and can be omitted.)

As an example, consider the graph $G = (W \cup X, E)$, shown in Figure 4.1. $X = \{x_1, x_2, x_3\}$, $W = \{a, b, c, d, e, f\}$.

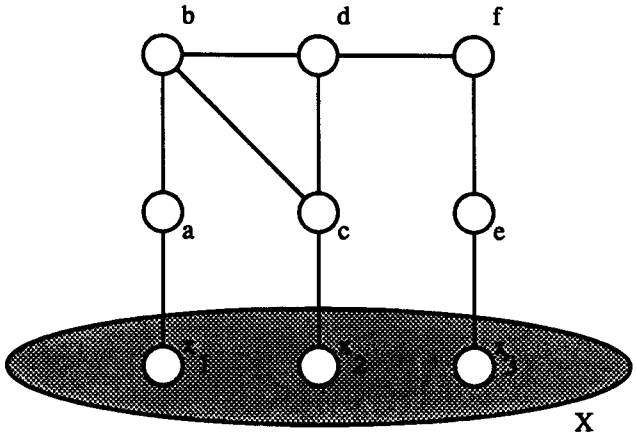

Figure 4.1

Here $P(W, X, x_1, x_2) = \{(1, \text{true}), (2, \text{false})\}$, because (x_1, a, b, d, c, x_2) is a play, where player 1 moves to x_2, and the resulting graph (with vertices d, f, e, x_3) gives a winning strategy for player 2, and (x, a, b, c, x_2) is a play where player 2 moves to x_2, and the resulting graphs gives a winning strategy for player 1. Similarly, $P(W, X, x_1, x_3) = \{(1, \text{true}), (2, \text{false})\}$. So $P(W, X, x_1) = \{(1, \text{true}, x_2), (2, \text{false}, x_2), (1, \text{true}, x_3), (2, \text{false}, x_3)\}$. For each subset R of $P(W, X, x_1)$ we can consider the game $VCC(X, W, x, R)$. Consider $R = \{(1, \text{true}, x_2), (2, \text{false}, x_2)\}$. Player 1 will win the game $VCC(X, W, x, R)$: player 2 must move from b. If he moves to d, player 1 moves to c, and the game ends with situation $(2, \text{false}, x_2)$. Otherwise, player 1 moves from c to x_2 and the game ends with situation $(1, \text{true}, x_2)$. As these are in R, player 1 wins. So f_1 in charac(X, W) has: $f_1(x_1, R) = \text{true}$. In total, f_1 can

be specified with $3 \cdot 2^8$ bits here. f_2 contains the information charac$(X - \{x_1\}, W)$, charac$(X - \{x_2\}, W)$ and charac$(X - \{x_3\}, W)$.

Note that if $|X|$ is bounded by some constant c, then the number of bits, needed to denote charac(X, W) is also bounded by some constant c', $|W|$ may be arbitrary large. (If S_k denotes this number for $|X| = k$, then $S_k \leq k \cdot 2^{(k-1)S_{k-2}} + k \cdot S_{k-1}$. So the constant factor grows very fast with k.)

For $i \in I$, let $Y_i = \{v \in X_j \mid j \text{ is a descendant of } i \text{ in } T\} - X_i$. Our algorithm is based upon computing charac(X_i, Y_i) for all $i \in I$.

Lemma 4.1
Let $i \in I$ be a leaf of T. Then $Y_i = \emptyset$, and charac(X_i, Y_i) can be computed in $\mathcal{O}(1)$ time.

Proof.
Clearly $Y_i = \emptyset$. Note that $|X_i| \leq k + 1 = \mathcal{O}(1)$. □

Lemma 4.2
There is a winning strategy for player 1 for VERTEX GENERALIZED GEOGRAPHY, if and only if charac(X_{i_0}, Y_{i_0}) = false.

Proof.
charac(X_{i_0}, Y_{i_0}) = charac$(\{s\}, V - \{s\})$ denotes whether there is a winning strategy for player 2 for VERTEX GENERALIZED GEOGRAPHY, played on $G[\{s\} \cup (V - \{s\})] = G$ with starting vertex s. □

Lemma 4.3
Let $i \in I$ be an internal node of T, and let j_1 and j_2 be the two children of i. Let charac(X_{j_1}, Y_{j_1}), charac(X_{j_2}, Y_{j_2}) be given. Then charac(X_i, Y_i) can be computed in $\mathcal{O}(1)$ time.

Proof.
Let $x \in X_i$. For all $R \subseteq P(X_i, Y_i, x)$, we can model all possible plays of the game $VCC(X_i, Y_i, x, R)$ by a rooted tree, as follows.

Nodes are of two types. One type of nodes is labeled with a 6-tupple $(v, p, Z_1, Z_2, c_1, c_2)$, where $v \in X_i \cup X_{j_1} \cup X_{j_2}$, $p \in \{1, 2\}$, $c_1 \in \bigcup_{Y \subseteq X_{j_1}} C(Y)$, $c_2 \in \bigcup_{Y \subseteq X_{j_2}} C(Y)$. This node represents the situation in the game, where player p must move from vertex v, and c_1 = charac(Z_1, W_1), Z_1 is the set of unvisited vertices in X_j, (but if $v \in X_{j_1}$, then $v \in Z_1$), and W_1 is the set of unvisited vertices in Y_{j_1}; c_2, Z_2 are defined in the same way with X_{j_2}, Y_{j_2}. The other type of nodes is labeled with a pair (α, R'), such that: the father of the node is labeled with $(v, p, Z_1, Z_2, c_1, c_2)$, and $v \in Z_\alpha$, $R' \subseteq P(Z_\alpha, W_\alpha, x)$, and there is a winning strategy for player 1 in the

game $VCC(Z_\alpha, W_\alpha, v, R')$ for W_α with charac$(Z_\alpha, W_\alpha) = C$. This corresponds to the situation that player p "decides to play the game $VCC(Z_\alpha, W_\alpha, v, R')$ ".

The root of the tree is labeled with (x, 1, X_{j_1}, X_{j_2}, charac(X_{j_1}, Y_{j_1}), charac(X_{j_2}, Y_{j_2})). A node labeled with (v, p, Z_1, Z_2, c_1, c_2) with $v \in X_i - \{x\}$ has no children, (because the game $VCC(X_i, Y_i, \alpha, R)$ ends when such a vertex is reached). A node, labeled with (v, p, Z_1, Z_2, c_1, c_2) with $v \notin X_i - \{x\}$ has children:

1. For all $w \in Z_1 \cup Z_2 \cup (X_i - \{x\} - X_{j_1} - X_{j_2})$ (i.e., all unvisited vertices in $X_i \cup X_{j_1} \cup X_{j_2}$) with $(v, w) \in E$, we take a childnode, labeled with (w, $3 - p$, Z_1', Z_2', c_1', c_2'), where $Z_1' = Z_1 - \{v\}$, $Z_2' = Z_2 - \{v\}$ (as v is now a vertex that has been visited). If $v \in Z_1$, then $c_1' =$ charac$(Z_1 - \{v\}, W_1)$ for some W_1 with $c_1 =$ charac(Z_1, W_1). Note that c_1' directly can be determined from c_1. (We do not need W_1). If $v \notin Z_1$, then $c_1' = c_1$. Similarly c_2' can be determined.

2. If $v \in Z_1$, then for every R' with $c_1 = (f_1, f_2)$ and $f_1(v, R') =$ true, i.e., where there is a winning strategy in $VCC(Z_1, W, v, R')$ for W with charac$(Z_1, W) = C_1$, there is a child-node, labeled with $(1, R')$.

3. If $v \in Z_2$, we take in the same way nodes, now labeled with $(2, R')$.

Next we consider nodes labeled with (α, R'). We suppose $\alpha = 1$. The case $\alpha = 2$ is similar. $(1, R')$ has a child-node for every $(p, c, y) \in R'$, labeled with (y, p', Z_1, Z_2, c, c_2), where p', Z_1, Z_2, c_2 are determined as follows. Let the father of the node, labeled with $(1, R')$ be labeled with (v, p'', Z_1', Z_2', c_1', c_2'). Then if $p = 1$, then $p' = 3 - p''$, and if $p = 2$, then $p' = p''$. (Player p'' acts as player 1 in a game $VCC(..., v, R')$. (p, c, y) is a possible ending of this game, where player p in this game moves to y. If $p = 1$, then this is player p'', so player $3 - p''$ must make a move from y, otherwise player p'' must move from y.) $Z_1 = Z_1' - \{v\}$, $Z_2 = Z_2' - \{v\}$. (v is the only vertex in $X_i \cup X_{j_1} \cup X_{j_2}$ that is visited during the "subgame" $VCC(..., v, R')$.) $c_2 =$ charac(Z_2, Y_{j_2}). (This is because no vertex in Y_{j_2} can be visited, when we start in X_i, go to vertices in Y_{j_1}, and stop when we reach a vertex in X_i — this follows from the definition of tree-decomposition.)

This finishes the description of the tree. It represents in an abstract way all possible plays of the game $VCC(X_i, Y_i, x, R)$. When vertices in Y_{j_1} or Y_{j_2} are visited, this is represented by considering "subgames" $VCC(Z, W, v, R')$. When the game is on a vertex $v \in X_i \cup X_{j_1} \cup X_{j_2}$, the state is denoted by a 6-tupple, giving basically: the vertex from which must be moved, the player that must move, the sets of unvisited vertices, and the "characteristics" of the subgraphs in Y_{j_1}, Y_{j_2} consisting of unvisited vertices.

We now show how to compute whether there is a winning strategy for player 1 in $VCC(X_i, Y_i, x, R)$ for $R \subseteq P(X, W, w)$. For each tree-node we determine whether it is "winning for player 1". There are a large number of different cases.

1. A node, labeled with (v, p, Z_1, Z_2, c_1, c_2) is a leaf of T, and $v \notin X_i$. Then this state is losing for player p: either he cannot make a move at all, or if he

moves to a vertex $\in Y_{j_1} \cup Y_{j_2}$, he will lose the game before the game reaches a vertex in $X_{j_1} \cup X_{j_2}$. (Note that $VCC(..., v, R)$ is lost here for all R, so also for R including all endings at a vertex in $Y_{j_1} \cup Y_{j_2}$).

2. For nodes, that are a leaf of T, and are labeled with (v, p, Z_1, Z_2, c_1, c_2) with $v \in X_i$, the characteristic of the resulting graph is uniquely determined by c_1, c_2 (and the structure of X_i, X_{j_1}, X_{j_2} and edges between these, and x and v). It is an element of $C(X - \{x, v\})$. It can be determined with the same procedure as described here. The recursion depth will be $\mathcal{O}(k)$, which is constant. (Using some tabulation of previous obtained results will help here to reduce the constant factor considerably). Suppose the resulting characteristic is c. Then the node corresponds to a state, that is winning for player p, if and only if $(v, p, c) \in R$. (Player p moves to v and the characteristic of the resulting graph is c.)

3. Consider a leaf-node, labeled with (α, R). Because the node is a leaf, $R = \emptyset$. Let the father of the node be labeled with (v, p, Z_1, Z_2, c_1, c_2). The node is winning for player p, because player p has a strategy that wins the game before any player moves to $Z_1 \cup Z_2$.

4. An internal node, labeled with (v, p, Z_1, Z_2, c_1, c_2) is winning for player p, if and only if at least one child of the node is winning for player p.

5. An internal node, labeled with (α, R'), with its father labeled with (v, p, Z_1, Z_2, c_1, c_2) is winning for player $3 - p$, if and only if at least one child of the node is winning for player p. (Player p decides to play a certain subgame $VCC(..., v, R)$. Player $3 - p$ can choose the actual outcome (p', c, y) of this subgame.)

In this way one can determine whether the root of the tree is winning for player 1, i.e., whether there is a winning strategy for player 1 in $VCC(X_i, Y_i, x, R)$.

All information, needed for charac(X_i, Y_i) can be determined in this way. As only charac(X_{j_1}), charac(X_{j_2}), and the structure of X_i, X_{j_1}, X_{j_2} and edges between these vertices are consulted, the procedure uses constant time. □

Theorem 4.4
For every constant $k \geq 1$: VERTEX GENERALIZED GEOGRAPHY (with specified starting vertex) can be solved in $\mathcal{O}(n)$ time for graphs $G = (V, E)$ with treewidth $\leq k$, that are given together with a tree-decomposition with treewidth $\leq k$.

Proof.
Compute for every $i \in I$ charac(X_i, Y_i). This is done by starting at leaf nodes (Lemma 4.1); and then repeatedly computing charac(X_i, Y_i) when this has been

computed for both children of i (Lemma 4.3). When charac(X_{i_0}, Y_{i_0}) has been determined, the answer of the problem can be given (Lemma 4.2). As per node $i \in I$ only constant time is used, this takes in total $\mathcal{O}(|I|) = \mathcal{O}(n)$ time. □

Note that the constant factor in the algorithm grows very fast with k. Basically, adding 2 to k gives one level of exponentiation in the constant factor extra. Thus our algorithm will only be practical for very small values of k, probably only for $k = 1, 2, 3$ and with some extra optimizations perhaps for $k = 4$ and 5.

It is possible to modify the algorithm in order to obtain similar results for related games.

Theorem 4.5
For every constant $k \geq 1$: VERTEX GENERALIZED GEOGRAPHY without specified starting vertex can be solved in $\mathcal{O}(n)$ time for graphs $G = (V, E)$ with treewidth $\leq k$, that are given together with a tree-decomposition with treewidth $\leq k$.

Proof.
VERTEX GENERALIZED GEOGRAPHY without starting vertex on $G = (V, E)$ is equivalent to VERTEX GENERALIZED GEOGRAPHY on $G' = (V \cup \{v^+, v^{++}\}, E \cup \{(v^+, v^{++})\} \cup \{(v^{++}, w) \mid w \in V\})$ with starting vertex v^+. It is easy to make a tree-decomposition of G' with treewidth $\leq k + 1$, given a tree-decomposition of G with treewidth $\leq k$. □

Theorem 4.6
For every constant $k \geq 1$: HAMILTONIAN CIRCUIT CONSTRUCTION GAME and HAMILTONIAN PATH CONSTRUCTION GAME can be solved in $\mathcal{O}(n)$ time for graphs $G = (V, E)$ with treewidth $\leq k$, that are given together with a tree-decomposition with treewidth $\leq k$.

Proof.
This is done with a method, similar to VERTEX GENERALIZED GEOGRAPHY. Basically, one must change the charac-functions a little, and incorporate in functions charac(X_i, W_i) whether $W_i = \emptyset$. We omit the details. □

It is not clear whether the other problems, that are considered on this paper can be solved in polynomial time on graphs with bounded treewidth. For the SIMPLE (ELEMENTARY) PATH CONSTRUCTION GAME, it seems that incorporating the length of the paths in the characteristics will give rise to characteristics of non-polynomial size. For EDGE GENERALIZED GEOGRAPHY, the size of the characteristics in our type of scheme becomes exponential, because a vertex v can be visited $\mathcal{O}(\text{degree}(v))$ times, which can be linear. This problem disappears, when we assume a fixed upper bound on the degree of the vertices. We use the following lemma.

Lemma 4.7
Let $G = (V, E)$ be a graph with treewidth $\leq k$ and maximum vertex degree $\leq d$. Then the treewidth of the edge graph of G is at most $(k+1)d - 1$.

Proof.
Consider a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G with treewidth $\leq k$. Take $Y_i = \{(v, w) \in E \mid v \in X_i \vee w \in X_i\}$. For every pair of edges (v, w), $(w, x) \in E$ note that $\exists i : (v, w), (w, x) \in Y_i$, namely take $i \in I$ with $w \in X_i$. Also note that the set of nodes $i \in I$ with $(v, w) \in Y_i$ is the union of the subtree of T $\{i \in I \mid v \in X_i\}$ and the subtree of T $\{i \in I \mid w \in X_i\}$. As $\exists i : v, w \in X_i$, these subtrees are not disjoint, hence their union is a connected subtree of T. It follows that $(\{Y_i \mid i \in I\}, T = (I, F))$ is a tree-decomposition of the edge-graph of G. Clearly $\forall i \in I : |Y_i| \leq d \cdot |X_i| \leq (k+1)d$. □

Theorem 4.8
For every constant $k \geq 1$, $d \geq 1$: EDGE GENERALIZED GEOGRAPHY can be solved in $\mathcal{O}(n)$ time for graphs $G = (V, E)$ with treewidth $\leq k$ and maximum vertex degree $\leq d$, that are given together with a tree-decomposition with treewidth $\leq k$.

Proof.
EDGE GENERALIZED GEOGRAPHY on G with starting vertex v is equivalent to VERTEX GENERALIZED GEOGRAPHY on the edge graph of G, with starting vertex one of the $\leq d$ vertices that correspond to an edge with head v. Now use Lemma 4.7 and Theorem 4.4. □

Clearly, Theorem 4.6 and 4.8 hold also for the case without specified starting vertex. We now consider an application to QUANTIFIED SATISFIABILITY.

Definition.
Let $F = Q_1 x_1 Q_2 x_2 \cdots Q_n x_n E$ be a well-formed quantified boolean expression, where each Q_i is either "$\forall$" or "$\exists$", and E is a boolean expression in conjunctive normal form. The graph G_F is defined as follows: $G_F = (\{x_1, \cdots, x_n\}, E_F)$ with $E_F = \{(x_i, x_{i+1}) \mid 1 \leq i \leq n\} \cup \{(x_i, x_j) \mid$ there exists a clause c in expression E, that contains a literal x_i or $\overline{x_i}$, and that contains a literal x_j or $\overline{x_j}\}$.

Corollary 4.9
One can decide in $\mathcal{O}(n)$ time whether a formula F of the form described above, is true, when the treewidth of G_F is bounded by a constant k, and E is given with a tree-decomposition of G_F with treewidth $\leq k$.

Proof.
Look to the transformation from QUANTIFIED 3-SATISFIABILITY to VERTEX GENERALIZED GEOGRAPHY, given in [29]. It is not hard to see that if G_F has

treewidth $\leq k$, then the treewidth of the graph, resulting from this transformation has treewidth $\leq \mathcal{O}(k)$, and that the corresponding tree-decomposition can be constructed from the tree-decomposition of G_F in $\mathcal{O}(n)$ time. Then apply Theorem 4.4. □

In all cases, if the required tree-decomposition is not given, it can be found (if it exists) in $\mathcal{O}(n^2)$ time [11, 32]. It is also possible to find parallel algorithms that use polylogarithmic time for the considered problems on graphs with bounded treewidth.

Theorem 4.10
For every constant $k \geq 1$: VERTEX GENERALIZED GEOGRAPHY, EDGE GENERALIZED GEOGRAPHY restricted to graphs with maximum degree $\geq d$ (d constant), HAMILTONIAN CIRCUIT CONSTRUCTION GAME, HAMILTONIAN PATH CONSTRUCTION GAME, when restricted to graphs with treewidth $\leq k$ belong to the class NC.

Proof.
This follows directly from the algorithms and the fact that a suitable tree-decomposition with T a tree of logarithmic depth can be found in polylogarithmic time on a (CRCW or EREW) PRAM [12]. □

A similar type of result holds for QUANTIFIED SATISFIABILITY. Lengauer introduced a method for hierarchical descriptions of graphs. With this method it is possible to specify graphs that have a size, exponential in the size of the specification. (See e.g. [27, 28].)

Theorem 4.11
For each constant $k \geq 1$: VERTEX GENERALIZED GEOGRAPHY, HAMILTONIAN PATH CONSTRUCTION GAME and HAMILTONIAN CIRCUIT CONSTRUCTION GAME for hierarchical graphs, where each cell contains at most k vertices, can be solved in time, linear in the size of the graphs.

Proof.
Use a method, similar to the method for graphs with bounded treewidth. For each cell G_i we compute charac(V_i, X_i), where V_i are the vertices in cell G_i, and X_i is the set of all other vertices in the expansion of G_i. We omit the details. □

4.2 EDGE GENERALIZED GEOGRAPHY on cacti

In this section we give a linear time algorithm for EDGE GENERALIZED GEOGRAPHY on cacti, based upon graph rewriting. The resulting algorithm is easier and more practical than the algorithm for EDGE GENERALIZED GEOGRAPHY on graphs with bounded treewidth and degree. Also we do not need to restrict the

degree of the graphs here, but on the other hand, the class of cacti is much more limited than that of the graphs with treewidth ≤ 2.

We will use the following notations: (G, s) denotes the game (instance), where the EDGE GENERALIZED GEOGRAPHY game is played on the graph G with starting vertex s. In the undirected graphs we deal with, we allow self-loops, and multiple edges. Write $(G, s) \equiv (H, t)$ if and only if there is a winning strategy for player 1 in $(G, s) \Leftrightarrow$ there is a winning strategy for player 1 in (H, t)).

Lemma 4.12
Let $G = (V, E)$ be an undirected graph. Let $s, v \in V$, degree$(v) = 1$, $(v, w) \in E$, $w \neq s$. Let $G - \{v, w\}$ denote the graph $(V - \{v, w\}, E - \{(x, y) \in E \mid x = v \vee x = w\})$. Then $(G, s) \equiv (G - \{v, w\}, s)$.

Proof.
Suppose player $j \in \{1, 2\}$ has a winning strategy in $(G - \{v, w\}, s)$. Then he has a winning strategy in (G, s) : as long as player $3 - j$ does not move to w, make the same moves as in $(G - \{v, w\}, s)$. When player $3 - j$ moves to w, then move to v and win the game. Now the lemma follows. □

Lemma 4.13
Let $G = (V, E)$ be an undirected graph. Let $(v, w) \in E$; degree(v) = degree$(w) = 2$; and $s \notin \{v, w\}$. Let x be the neighbor of $v \neq w$, and let y be the neighbor of $w \neq v$. Let $G' = (V - \{v, w\}, E - \{(x, v), (v, w), (w, y)\} \cup \{(x, y)\})$. Then $(G, s) \equiv (G', s)$.

Proof.

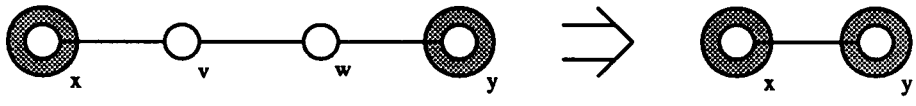

Figure 4.2

The construction is shown in Figure 4.2. The game on both graphs is similar, as e.g. moving from x to y in G' corresponds in moving from x to v, then the other player will move to w, and then the player moves to y. □

Lemma 4.14
Let $G = (V, E)$ be a directed graph. Let $v \in V$; degree$(v) = 2$; and suppose both edges adjacent to v have the same other endpoint w. Suppose $v \neq s$. Let $G - \{v\} = G[V - \{v\}]$. Then $(G, s) \equiv (G - \{v\}, s)$.

Proof.
The construction is shown in Figure 4.3. If there exists a winning strategy in $(G - \{v\}, s)$ for player $j \in \{1, 2\}$, then there exists one in (G, s): move as in $(G - \{v\}, s)$, except when player $3 - j$ moves to v, then move back to w. □

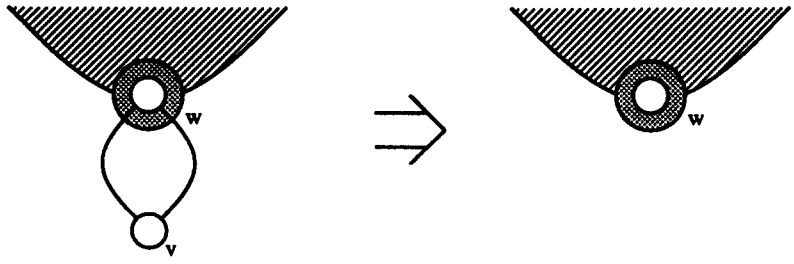

Figure 4.3

Lemma 4.15
Let $G = (V, E)$ be an undirected graph. Let $v \in V$ be adjacent to two self-loops $e_1 = (v, v)$ and $e_2 = (v, v)$ $(e_1 \neq e_2)$. Then $(G, s) \equiv (G', s)$ with $G' = (V, E - \{e_1, e_2\})$.

Proof.
Similar as before. When player $3 - j$ moves over e_1 or e_2, then player j moves over the other edge in $\{e_1, e_2\}$. (See Figure 4.4 for the construction.) □

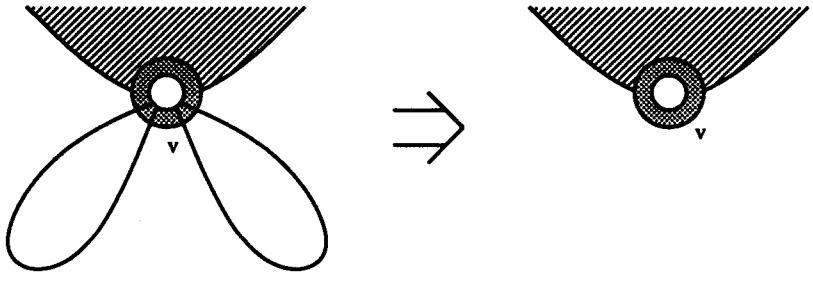

Figure 4.4

Lemma 4.16
Let $G = (V, E)$ be an undirected graph. Let $v \in V$; suppose v is adjacent to exactly 3 edges, where exactly one of these is a self-loop; and suppose $v \neq s$. Let $G - \{v\}$ be as above. Then $(G, s) \equiv (G - \{v\}, s)$.

Proof.
Suppose there is a winning strategy for player $j \in \{1,2\}$ in $(G-\{v\},s)$. Then there is a winning strategy for player j in (G,s). As long as player $3-j$ does not move to v, player j moves as in $(G-\{v\},s)$. Suppose player $3-j$ moves to v. Let e_1 be the selfloop (v,v), and let e_2 be the other unused edge, adjacent to v. (See Figure 4.5.) Moving over edge e_2 is either a winning or a losing move, regardless what player makes the move. So if it is a winning move, player j moves over e_2, and if it is a losing move, then player j moves over e_1, and player $3-j$ must move over e_2 and loses the game. (See Figure 4.6 for the construction.) □

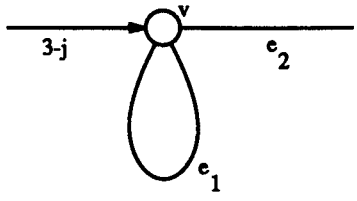

Figure 4.5

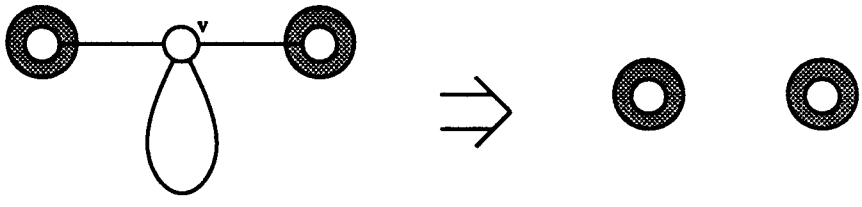

Figure 4.6

Lemma 4.17
Let $G=(V,E)$ be an undirected graph. Let $v \in V$; suppose v is adjacent to exactly 2 edges, one of which is a self-loop. Let $(v,w) \in E$, $v \neq w$ be the other edge. Suppose $v \neq s$. Then $(G,s) \equiv (G-\{v\},s)$.

Proof.
Similar as before. Player j plays in G as in $G-\{v\}$, but when player $3-j$ moves to v from w, then player j moves over the self-loop and wins the game. (See Figure 4.7 for the construction.) □

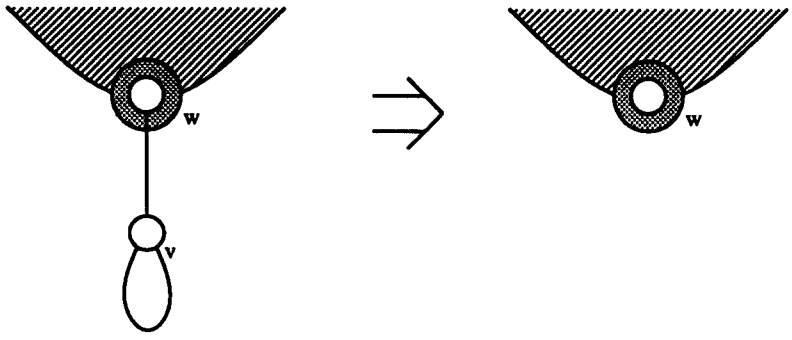

Figure 4.7

Lemma 4.18
After applying the rules of Lemma 4.12 – 4.17 as often as possible, starting with (G, s) with G a connected cactus, a game (H, s) will result, with $H = (\{s\}, \emptyset)$; $H = (\{s\}, \{(s, s)\})$ or $H = (\{s, v\}, \{(s, v)\})$ for some v.

Proof.
(See Figure 4.8 for the possibilities for H.) Each application of a Lemma 4.12 – 4.17 will result in another, smaller cactus. Note that the biconnected components of G form a tree. Every leaf-node in this tree that corresponds to a single edge or a cycle with even length will disappear with Lemma 4.12, 4.13 and 4.14. Every leaf-node corresponding to a cycle with odd length will reduce to a self-loop. Suppose no application of a Lemma 4.12 – 4.16 is possible. The resulting graph H cannot have more than one biconnected component. Suppose not. Look at a biconnected component that is a leaf in the tree of biconnected components if we do not look to self-loops. If it is a cycle, and some vertices ($\neq$ the unique vertex, adjacent to other biconnected components) have self-loops, then Lemma 4.15 or 4.16 can be applied. If it is a cycle without such self-loops, it can be reduced to nothing or a self-loop. If it is a single edge, then Lemma 4.12, 4.15 or 4.17 can be applied. With a similar argument, H cannot have a single biconnected component with three or more vertices. So H has at most 2 vertices. Simple case analysis gives the theorem. □

Theorem 4.19
EDGE GENERALIZED GEOGRAPHY can be solved in $\mathcal{O}(n)$ time on cacti.

Proof.
First we remark that we may restrict ourselves to connected graphs. Cacti have $\mathcal{O}(n)$ edges. It remains to show that by proper choice of data-structures, we can dynamically determine where one of the Lemmas 4.12 – 4.17 can be applied, in

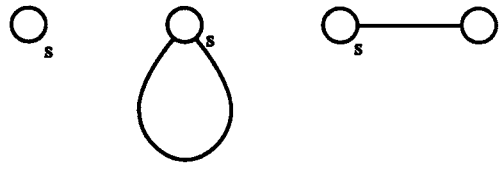

Figure 4.8

$\mathcal{O}(1)$ amortized time per operation. Hereto, each vertex has a counter, denoting its degree, and a boolean, denoting whether it has a self-loop. (We may assume, by Lemma 4.15 that each vertex has 0 or 1 adjacent self-loops.) In a queue Q we put each vertex, where one of the rules 4.12 – 4.14, 4.16, 4.17 can be applied. Repeatedly, a vertex is taken from Q; if the vertex has not been deleted already by an earlier operation, the operation corresponding to v is applied; for each removed edge its still existing endpoints have their degree updated, and are possibly put in Q, and some other checks are made (depending on the particular operation), possibly resulting in the setting of a "self-loop boolean", or putting one or more vertices in Q. We omit the easy, but tedious details. Finally, if the resulting graph $H = (\{s\}, \emptyset)$, then player 2 has a winning strategy, otherwise player 1 has a winning strategy. □

It is possible to prove other lemmas, of a similar flavor as Lemmas 4.12 – 4.17. With similar techniques one can show:

Theorem 4.20
EDGE GENERALIZED GEOGRAPHY can be solved in $\mathcal{O}(n)$ times for directed graphs $G = (V, E)$, with the property that the undirected graph $G' = (V, \{(v, w) \mid (v, w) \in E \vee (w, v) \in E\})$ is a cactus.

Also, similar algorithms can be designed for the VERTEX GENERALIZED GEOGRAPHY game on cacti.

5 Final comments.

This research leaves several directions for further research. On one hand, there are still several interesting variants, that have not yet been shown to be PSPACE-complete, like EDGE GENERALIZED GEOGRAPHY without specified starting vertex, and most of the games, considered in this paper, on undirected graphs. We discuss one of these games later in this section. On the other hand, much work can still be done on the complexity of the problems, when restricted to special classes of graphs. It is surprising to contrast the little amount of work done on special cases of PSPACE-complete problems with the huge amount of work done on special cases of NP-complete (graph) problems. (See e.g. [23, 25]). Although results for special

cases of PSPACE-complete problems usually will be harder to obtain than similar results for NP-complete problems, there are many interesting problems in this area, that are worth being studied.

We now consider VERTEX GENERALIZED GEOGRAPHY on undirected graphs. A possible PSPACE-hardness proof for this problem could use a transformation from VERTEX GENERALIZED GEOGRAPHY in the following way: each edge (v, w) is replaced by a graph $G_{OW} = (V, E)$ with two specified vertices $s, t \in V$. s is identified with v, t is identified with w. s and t have degree 1 in G_{OW}. (See Figure 5.1.)

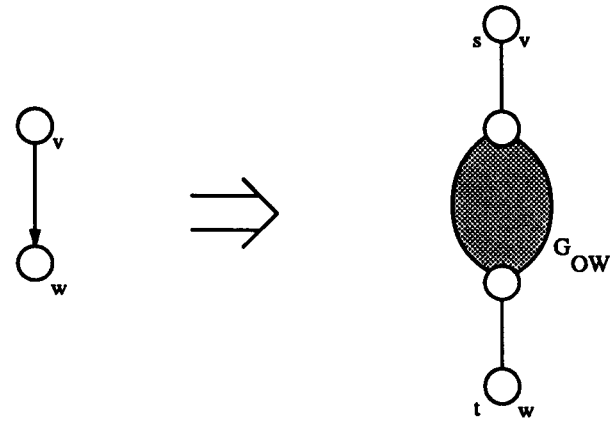

Figure 5.1

Consider the following variant of VERTEX GENERALIZED GEOGRAPHY. The game has starting vertex s (t). A player that moves to t (s) does not win the game, but scores "a kind of" *draw*. Use the following scoring values. A player that cannot move from a vertex $\neq t(s)$ scores 0, its opponent 1. The player that moves to t scores 1/3, its opponent 2/3.

Suppose G_{OW}, s, t fulfill the following two requirements:

1. There is a strategy for player 1 in the game with starting vertex t and "drawing" vertex s, that guarantees player 1 to score 1 point.

2. There is a strategy for player 1 in the game with starting vertex s and "drawing" vertex t, that guarantees player 1 to score 1/3 point, but there is no strategy for player 1 that guarantees to score 2/3 or 1 point.

We call such a graph G_{OW} a "one way" graph.

Lemma 5.1
Suppose there exists an (undirected) one way graph. Then VERTEX GENERALIZED GEOGRAPHY is PSPACE-complete for undirected graphs.

Proof.
Clearly the problem is in PSPACE. To prove PSPACE-hardness under the assumption that an undirected one way graph G_{OW} exists we transform from the directed case. Let an instance $G = (V, E)$, s_0 of VERTEX GENERALIZED GEOGRAPHY for directed graphs been given. Replace each edge (v, w) by a copy of G_{OW}, identifying v with s and w with t. Now note the following: 1. When $(v, w) \in E$, then a player that moves from w to a vertex in the copy of G_{OW}, corresponding to edge (v, w) will lose the game. 2. When $(v, w) \in E$ and player i moves from v to a vertex in the copy of G_{OW}, corresponding to (v, w), then for both players there is a strategy that will result in the situation that player $3-i$ must move from w. This means that the undirected one way graph G_{OW} functions in the same way as a directed edge. Hence, there is a winning strategy for player 1 in the obtained undirected graph, if and only if there is one in G, s_0. As the transformation can be done in logarithmic working space, the theorem follows. □

Hence, a very interesting open problem is the following.

Question: Does there exist an undirected one way graph?

Despite several investigations (including automatically testing a large number of randomly generated undirected graphs) we were unable to find an undirected one way graph, and unable to disprove its existence. A similar approach could also be tried for the EDGE GENERALIZED GEOGRAPHY problem on undirected graphs. This seems a harder problem (consider the proof of Theorem 4.8).

We close with mentioning without proof some other special cases of the problems considered in this paper:

- VERTEX GENERALIZED GEOGRAPHY, EDGE GENERALIZED GEOGRAPHY, SIMPLE PATH CONSTRUCTION GAME and ELEMENTARY PATH CONSTRUCTION GAME are solvable in $\mathcal{O}(n + e)$ time on acyclic graphs, but become PSPACE-complete if restricted to graphs, obtained by adding one edge to an acyclic graph.
- All games, that are considered in this paper, except for TRON without specified starting vertices, are linear time solvable on (undirected graphs that are) trees. Recently, an $\mathcal{O}(n\sqrt{n})$ algorithm for TRON without specified starting vertices was obtained together with Kloks [15].
- SIMPLE PATH CONSTRUCTION GAME is solvable in $\mathcal{O}(n)$ time on cacti.
- ELEMENTARY PATH CONSTRUCTION GAME is solvable in $\mathcal{O}(n^3 + 2^{d/2}n)$ time for cacti with maximum vertex degree d.

Acknowledgement

Marinus Veldhorst proposed the game, called TRON in this paper.

References

[1] A. Adachi, S. Iwata, and T. Kasai. Some combinatorial game problems require $\Omega(n^k)$ time. *J. ACM*, 31:361–376, 1984.

[2] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[4] S. Arnborg, J. Lagergren, and D. Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Proc. 15 th ICALP*, pages 38–51. Springer Verlag, Lect. Notes in Comp. Sc. 317, 1988. To appear in J. of Algorithms.

[5] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.

[6] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Disc. Appl. Math.*, 23:11–24, 1989.

[7] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8:216–235, 1987.

[8] H. L. Bodlaender. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Dept. of Computer Science, Utrecht University, Utrecht, 1986.

[9] H. L. Bodlaender. Dynamic programming algorithms on graphs with bounded tree-width. Technical report, Lab. for Computer Science, M.I.T., 1987. Ext. abstract in proceedings ICALP 88.

[10] H. L. Bodlaender. Polynomial algorithms for Chromatic Index and Graph Isomorphism on partial k-trees. Technical Report RUU-CS-87-17, Dept. of Computer Science, Utrecht University, 1987.

[11] H. L. Bodlaender. Improved self-reduction algorithms for graphs with bounded treewidth. Technical Report RUU-CS-88-29, Dept. of Computer Science, Utrecht University, 1988. To appear in: Proc. Workshop on Graph Theoretic Concepts in Comp. Sc. '89.

[12] H. L. Bodlaender. NC-algorithms for graphs with small treewidth. In J. van Leeuwen, editor, *Proc. Workshop on Graph-Theoretic Concepts in Computer Science WG'88*, pages 1–10. Springer Verlag, Lecture Notes in Computer Science vol. 344, 1988.

[13] H. L. Bodlaender. Planar graphs with bounded treewidth. Technical Report RUU-CS-88-14, Dept. of Computer Science, Utrecht University, Utrecht, 1988.

[14] H. L. Bodlaender. On the complexity of some coloring games. Tech. Rep. RUU-CS-89-27, Department of Computer Science, University of Utrecht, 1989.

[15] H. L. Bodlaender and T. Kloks. Deciding the TRON-game on trees in $o(n\sqrt{n})$ time. Unpublished manuscript, 1990.

[16] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursive constructed graph families. Manuscript, 1988.

[17] B. Courcelle. A representation of graphs by algebraic expressions and its use for graph rewriting systems. In *Proc. 3rd Int. Workshop on Graph Grammars*, pages 112–132. Springer Verlag Lecture Notes in Computer Science vol. 291, 1987.

[18] S. Even and R. Tarjan. A combinatorial problem which is complete in polynomial space. *J. ACM*, 23:710–719, 1976.

[19] M. R. Fellows and K. Abrahamson. Cutset-regularity beats well-quasi-ordering for bounded treewidth. Manuscript, 1989.

[20] A. Fraenkel, M. Garey, D. Johnson, T. Schaefer, and Y. Yesha. The complexity of checkers on an $n \times n$ board — preliminary version. In *Proc. 19th Annual Symp. on Foundations of Computer Science*, pages 55–64, 1978.

[21] A. Fraenkel and E. Goldschmidt. Pspace-hardness of some combinatorial games. *J. Combinatorial Theory ser. A*, 46:21–38, 1987.

[22] A. Fraenkel and D. Lichtenstein. Computing a perfect strategy for n by n chess requires time exponential in n. *J. Combinatorial Theory ser. A*, 31:199–214, 1981.

[23] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[24] D. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 4:397–411, 1983.

[25] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 6:434–451, 1985.

[26] C. Lautemann. Efficient algorithms on context-free graph languages. In *Proc. 15'th ICALP*, pages 362–378. Springer Verlag, Lect. Notes in Comp. Sc. 317, 1988.

[27] T. Lengauer. Efficient algorithms for finding minimum spanning forests in hierarchically defined graphs. *J. Algorithms*, 8:260–284, 1987.

[28] T. Lengauer and E. Wanke. Efficient solutions of connectivity problems on hierarchically defined graphs. *SIAM J. Comput.*, 17:1063–1080, 1988.

[29] D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *J. ACM*, 27:393–401, 1980.

[30] J. Matoušek and R. Thomas. Algorithms finding tree-decompositions of graphs. Unpublished paper, 1988.

[31] K. Mehlhorn, S. Näher, and M. Rauch. On the complexity of a game related to the dictionary problem. In *Proceedings 30th Annual Symposium on Foundations of Computer Science*, pages 546–548, 1989.

[32] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. Manuscript, 1986.

[33] J. Robson. N by N Checkers is exptime complete. *SIAM J. Comput.*, 13:252–267, 1984.

[34] J. Robson. Alternation with restrictions on looping. *Information and Control*, 67:2–11, 1985.

[35] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Symp. on Theory of Computation*, pages 216–226, 1978.

[36] T. J. Schaefer. On the complexity of some two-person perfect-information games. *J. Comp. Syst. Sc.*, 16:185–225, 1978.

[37] P. Scheffler. *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme.* PhD thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.

[38] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8:151–174, 1979.

[39] T. Wimer. *Linear algorithms on k-terminal graphs.* PhD thesis, Dept. of Computer Science, Clemson University, 1987.