Look-ahead algorithms can also be extended to handle some generalized definitions of consistency, such as relational consistency, described in Chapter 8. Those definitions focus on constraints rather than on variables and are particularly well-suited for the nonbinary case. The "generalized arc-consistency" condition defined in Chapter 3 is particularly appropriate since it reduces domains only. You can try to extend SELECT-VALUE-ARC-CONSISTENCY and forward-checking to the generalized arc-consistency definition (see Exercise 2).

5.4 Satisfiability: Look-Ahead in Backtracking

The backtracking algorithm and its advances are applicable to the special case of propositional satisfiability. The most well-known, and still one of the best, variation of backtracking for this representation is known as the *Davis-Putnam, Logemann, and Loveland procedure* (DPLL) (Davis, Logemann, and Loveland 1962). This is a backtracking algorithm applied to a CNF theory (a set of clauses) augmented with unit propagation as the look-ahead method (see Figure 3.16 in Chapter 3). This level of look-ahead is akin to applying a type of full arc-consistency (i.e., relational arc-consistency) at each node.

Figure 5.13 presents DPLL. Unlike our general exposition of search algorithms, we define this one recursively. The algorithm systematically searches the space of

DPLL(φ)

Input: A CNF theory φ.

Output: A decision of whether φ is satisfiable.

1. Unit_propagate(φ);
2. **If** the empty clause is generated, return(*false*);
3. **Else**, if all variables are assigned, return(*true*);
4. **Else**
5. Q = some unassigned variable;
6. **return**(DPLL($\varphi \wedge Q$) $\vee$ DPLL($\varphi \wedge \neg Q$)) and if consistent, a truth assignment.

Figure 5.13 The DPLL procedure.

Dechter, Rina. Constraint Processing.
Morgan-Kaufmann, 2003

partial truth assignments of propositional variables. Step 1 applies unit propagation (line 1), accomplishing some level of dynamic variable and value selection automatically within unit propagation. Additional value and variable selection can be done at step 5 of Figure 5.13.

EXAMPLE 5.10 Consider the party problem with the following rules: If Alex attends the party, then Bill will as well. If Chris attends, then Alex will, and if both Alex and Bill do not attend, then David will. Assume we also know that Chris did attend the party. The CNF theory for this story is $\varphi = \{(\neg A \vee B), (\neg C \vee A), (A \vee B \vee D), (C)\}$. The search tree of our party example along ordering A, B, D, C is shown in Figure 5.14. We see that there are two consistent scenarios. Suppose we now apply the DPLL algorithm. Applying unit propagation will resolve unit clause C, yielding unit clause A, which in turn is resolved, yielding the unit clause B. Unit propagation terminates and we have three variable assignments $(\langle A = 1, B = 1, C = 1\rangle)$. In step 5 the algorithm will select the only variable D. Since both of its truth assignments are consistent, we have two solutions. In this case the algorithm encountered no dead-ends. The portion of the search space explored is enclosed in Figure 5.14. •

Higher levels of look-ahead using stronger resolution-based inference, which enforce higher levels of local consistency, can naturally be considered here. Different levels of bounded resolution, restricted by the size of the generated resolvents or by the size of the participating clauses, can be considered for propagation at each node in the search tree. The level of such propagation that is cost-effective is not clear, leading to the common trade-off between more inference at each node versus exploration of fewer nodes in the search tree. Clearly, when

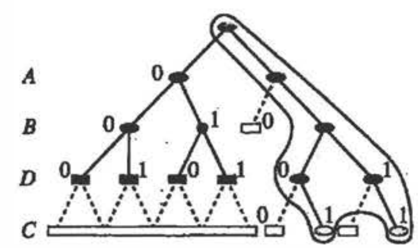

Figure 5.14 A state search tree along the variables A, B, D, C for a CNF theory $\varphi = \{(\neg A \vee B), (\neg C \vee A), (A \vee B \vee D), C\}$. Hollow nodes and bars in the search tree represent illegal states; gray ovals represent solutions. The enclosed area corresponds to DPLL with unit propagation when D is before C.