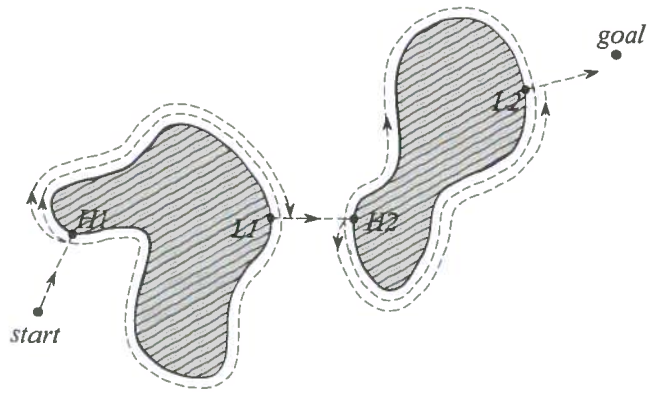

Figure 6.14
Bug1 algorithm with H1, H2, hit points, and L1, L2, leave points [199].

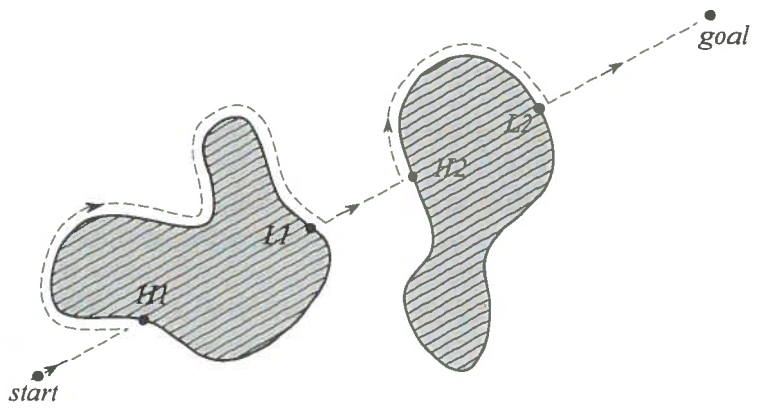

Figure 6.15
Bug2 algorithm with H1, H2, hit points, and L1, L2, leave points [199].

Practical application: example of Bug2. Because of the popularity and simplicity of Bug2, we present a specific example of obstacle avoidance using a variation of this technique. Consider the path taken by the robot in figure 6.15. One can characterize the robot's motion in terms of two states, one that involves moving toward the goal and a second that involves moving around the contour of an obstacle. We will call the former state GOAL-

SEEK and the latter WALLFOLLOW. If we can describe the motion of the robot as a function of its sensor values and the relative direction to the goal for each of these two states, and if we can describe when the robot should switch between them, then we will have a practical implementation of Bug2. The following pseudocode provides the highest-level control code for such a decomposition:

```
public void bug2(position goalPos){
 boolean atGoal = false;

 while( ! atGoal){
   position robotPos = robot.GetPos(&sonars);
   distance goalDist = getDistance(robotPos, goalPos);
  angle goalAngle = Math.atan2(goalPos, robotPos)-robot.GetAngle();
   velocity forwardVel, rotationVel;

   if(goalDist < atGoalThreshold){
     System.out.println("At Goal!");
     forwardVel = 0;
     rotationVel = 0;
     robot.SetState(DONE);
     atGoal = true;
   }
   else{
     forwardVel = ComputeTranslation(&sonars);
     if(robot.GetState() == GOALSEEK){
        rotationVel = ComputeGoalSeekRot(goalAngle);
        if(ObstaclesInWay(goalAngle, &sonars))
          robot.SetState(WALLFOLLOW);
     }
     if(robot.GetState() == WALLFOLLOW){
        rotationVel = ComputeRWFRot(&sonars);
        if( ! ObstaclesInWay(goalAngle, &sonars))
          robot.SetState(GOALSEEK);
     }
   }
   robot.SetVelocity(forwardVel, rotationVel);
 }
}
```

In the ideal case, when encountering an obstacle one would choose between left wall following and right wall following depending on which direction is more promising. In this simple example we have only right wall following, a simplification for didactic purposes that ought not find its way into a real mobile robot program.