Chapter 22 describes a model of Harald Zhang (1997) and Alvarez and Jermann (2000, 2001). The model introduces participation (collateral) constraints and shocks in a way that makes a changing subset of agents i satisfy (1.3.20). Zhang and Alvarez and Jermann formulate these models by adding participation constraints to the recursive formulation of the consumption problem based on (1.4.7). Next we briefly describe the structure of these models and their attitude toward our theme equation, the consumption Euler equation (1.3.3). The idea of Zhang and Alvarez and Jermann was to meet the empirical asset pricing challenges by disrupting (1.3.3). As we shall see, that requires eliminating some of the assets that some of the households can trade. These advanced models exploit a convenient method for representing and manipulating history dependence.

1.4. Recursive methods

The pervasiveness of the consumption Euler inequality will be a significant substantive theme of this book. We now turn to a methodological theme, the imperialism of the recursive method called dynamic programming.

The notion that underlies dynamic programming is a finite-dimensional object called the *state* that, from the point of view of current and future payoffs, completely summarizes the current situation of a decision maker. If an optimum problem has a low-dimensional state vector, immense simplifications follow. A recurring theme of modern macroeconomics and of this book is that finding an appropriate state vector is an art.

To illustrate the idea of the state in a simple setting, return to the savings problem and assume that the consumer's endowment process is a time-invariant function of a state s_t that follows a Markov process with time-invariant one-period transition density $\pi(s'|s)$ and initial density $\pi_0(s)$, so that $y_t = y(s_t)$. To begin, recall the description (1.3.5) of consumption that prevails in the special linear quadratic version of the savings problem. Under our present assumption that y_t is a time-invariant function of the Markov state, (1.3.5) and the household's budget constraint imply the following representation of the household's decision rule:

$$c_t = f(A_t, s_t) \tag{1.4.1a}$$

$$A_{t+1} = g(A_t, s_t). \tag{1.4.1b}$$

Equation (1.4.1a) represents consumption as a time-invariant function of a state vector (A_t, s_t). The Markov component s_t appears in (1.4.1a) because it contains all of the information that is useful in forecasting future endowments (for the linear quadratic model, (1.3.5) reveals the household's incentive to forecast future incomes), and the asset level A_t summarizes the individual's current financial wealth. The s component is assumed to be exogenous to the household's decisions and has a stochastic motion governed by $\pi(s'|s)$. But the future path of A is chosen by the household and is described by (1.4.1b). The system formed by (1.4.1) and the Markov transition density $\pi(s'|s)$ is said to be *recursive* because it expresses a current decision c_t as a function of the state and tells how to update the state. By iterating (1.4.1b), notice that A_{t+1} can be expressed as a function of the history $[s_t, s_{t-1}, \ldots, s_0]$ and A_0. The endogenous state variable financial wealth thus encodes all payoff-relevant aspects of the history of the exogenous component of the state s_t.

Define the value function $V(A_0, s_0)$ as the optimum value of the savings problem starting from initial state (A_0, s_0). The value function V satisfies the following functional equation, known as a Bellman equation:

$$V(A, s) = \max_{c, A'} \{u(c) + \beta E[V(A', s')|s]\} \tag{1.4.2}$$

where the maximization is subject to $A' = R(A+y-c)$ and $y = y(s)$. Associated with a solution $V(A, s)$ of the Bellman equation is the pair of policy functions

$$c = f(A, s) \tag{1.4.3a}$$

$$A' = g(A, s) \tag{1.4.3b}$$

from (1.4.1). The *ex ante* value (i.e., the value of (1.3.1) before s_0 is drawn) of the savings problem is then

$$v(A_0) = \sum_s V(A_0, s)\, \pi_0(s). \tag{1.4.4}$$

We shall make ample use of the *ex ante* value function.