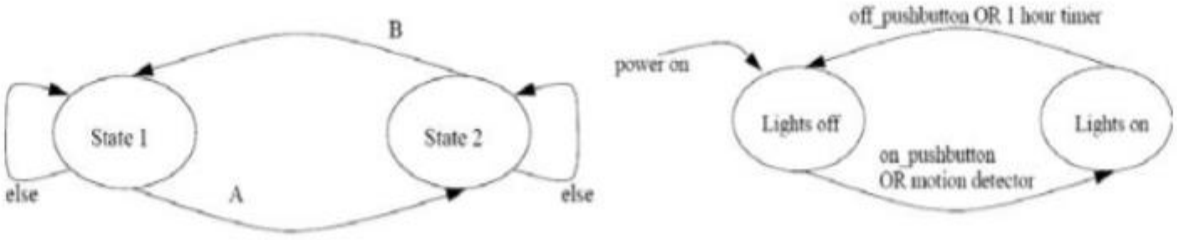
Metod konačnih automata

Većina sistema u oblasti automatskog upravljanja su sekvencijalni po svojoj prirodi. Tokom noramalnog rada, sekvencijalni sistem prolazi kroz više koraka, tj. faza ili stanja. U svakom stanju, sistem se ponaša na drugačiji način. Na primer, zamislimo semafor za regulisanje saobraćaja. Jedno stanje semafora može biti ono kada je dozvoljen saobraćaj u jednom, a drugo kada je dozvoljen saobraćaj u drugom smeru. Svakom stanju semafora odgovara jedna specifična kombinacija upaljenih crvenih, žutih i zelenih svetala, a stanja se izmenjuju po unapred utvrđenom redosledu, tj. sekvenci. Ova sekvenca može biti fiksna (nepromenljiva), ali se može i menjati pod dejstvom nekih specifičnih ulaznih uslova. Na primer, ako na semaforu postoji ″pešačko dugme″, tada će redosled u kome se izmenjuju semaforska svetla svakako zavisiti i od toga da li je dugme pritisnuto ili ne. Dakle, pod stanjem se može smatrati vremenski interval u kojem su izlazi sistema stabilni (ne menjaju se) i u kome sistem reaguje na ulaznu pobudu na način specifičan za to stanje.

Sekvencijalni sistemi su po pravilu složeni, a njihovo projektovanje je otežano obiljem detalja o kojima treba voditi računa. Zbog toga, direktno projektovanje, gde se na osnovu polazne specifikacije, a bez neke veće razrade, odmah pristupa implementaciji (pisanju programa, ili crtanju leder dijagrama) ima ograničen domet, a dobijena rešenja često nisu ispravna ili zahtevaju naknadno temeljno testiranje i ispravljanje učinjenih grašaka. Iz tog razloga, razvijeno je više, u osnovi srodnih, metoda projektovanja sekvencijalnih sistema kod kojih je naglasak upravo na razradi problema sa ciljem da se, pre same implementacije, detaljnom analizom identifikuju karakteristike sistema, kao što su stanja, prelazi i događaji, na osnovu kojih se kreira apstraktni model ponašanja sistema koji se potom koristi za neposrednu implementaciju. Jedan od takvih metoda je upravo metod konačnih automata. Ovaj metod se primenjuje ne samo za opis ponašanja sistema automatskog upravljanja, već i u mnogim drugim oblastima, kao što je projektovanje upravljačkih jedinica digitalnih sistema, ili projektovanje konkurentnog softvera.

Model konačnog automata zasnovan je na konceptu stanja i prelaza između stanja. U svakom trenutku, automat je u jednom od konačnog broja svojih stanja. Pod dejstvom događaja ili uslova, a u zavisnosti od tekućeg stanja i trenutne vrednosti ulaza, automat prelazi u novo stanje i generiše odgovaraju će izlaze. Konačni automat se može predstaviti dijagramom (grafom) stanja. U ovom dijagramu, krugovima su predstavljena stanja, a strelicama (granama) prelazi između stanja.

Dijagram stanja prikazan na Sl. 4-17(a) ima dva stanja State 1 i State 2. Ako je sistem u stanju State 1 i desi se događaj A, sistem prelazi u stanje State 2, inače ostaje u stanju State 1. Slično, ako je sistem u stanju State 2 i desi se događaj B, sistem se vraća u stanje State 1, inače ostaje u stanju State 2. Grane na Sl. 4-17(a) označene sa else važe u slučajevima kada ni jedan uslov naveden na izlaznim granama iz dataog stanja nije ispunjen. Uobičajeno je da se ove grane ne crtaju, već se podrazumevaju, tj. ako ni jedan uslov koji vodi sistem u neko drugo stanje nije ispunjen, sistem ostaje u zatečenom stanju.



(a) (b)

Sl. 4-17. (a) Dijagram stanja sa dva stanja; (b) dijagram stanja kontrola osvetljenja.

Dijagram stanja sa Sl. 4-17(a) može se iskoristiti za modelovanje ponašanja kontrolera osvetljenja, kao što je prikazano na Sl. 4-17(b). Stanjima su sada data imena koja ukazuju na režim rada sistema. U stanju Light off svetlo je ugašeno, a u stanju Ligth on svetlo je upaljeno. Događaj koji prevodi sistem iz stanja Light off u stanje Light on je ″prekidač uključen ili detektovan pokret″. Događaj koji gasi svetlo, tj. prevodi sistem iz stanja Ligth on u stanje Light off je ″isteklo vreme od 1h ili prekidač isključen″. Strelica usmerena ka stanju Light off i označena sa reset ukazuje na početno stanje sistema, tj. na stanje u koje će sistem biti postavljen kada počne sa radom.