Bevezetés

A λ-kalkulus első definícióját és elméletének első eredményeit Church adta meg 1932–33-ban. Church elsődleges célja az volt, hogy a teljes matematika formális leírására adjon egy rendszert, de nem sokkal később kiderült, hogy rendszerével ellentmondások is leírhatók, így ez nem volt megfelelő az eredeti cél megvalósítására.

Church λ-kalkulusnak nevezett elméletével azonban a függvények tulajdonságai jól vizsgálhatók, és a λ-kalkulus volt az első olyan „eszköz”, amellyel a függvények kiszámíthatósága kezelhető lett.

A λ-kalkulus rendkívül egyszerű, és használhatóságát az is adja, hogy a rekurzió benne a rekurzió explicit leírása nélkül írható le. A λ-kalkulus egy konzisztens matematikai rendszer, nincs benne ellentmondás, sem paradoxonok.

Kleene megmutatta, hogy minden kiszámítható függvény leírható λ-kalkulusban, és a λ-definiálható függvények pontosan a kiszámítható függvények. A Turing-tétel a λ-definiálhatóság és a Turing-kiszámíthatóság ekvivalenciáját adta. A Church–Rosser-tételek pedig kimondták, hogy egy λ-kifejezés értékének a meghatározásakor a számítás eredménye, ha létezik, független a kiszámításban alkalmazott stratégiától.

A λ-kalkulus az első funkcionális programnyelvnek tekinthető, annak ellenére, hogy kidolgozásának időpontjában még nem is voltak számítógépek. Ugyanakkor a λ-kalkulus egy olyan egyszerű funkcionális programnyelv, amelyre minden más „magas szintű” funkcionális nyelven írt program átalakítható. Ezért elegendő a λ-kalkulust interpretálni, egy λ-kalkulus-interpreter minden funkcionális nyelvű programot végrehajthat.

Minden funkcionális program egy λ-kifejezésnek tekinthető. A kifejezések egyszerűbb leírására a funkcionális nyelvekben függvénydefiníciók adhatók meg, a funkcionális nyelven megírt programban ezekre a függvényekre a definíciókban megadott függvénynevekkel lehet hivatkozni. A λ-kalkulusban erre nincs lehető-

ség, és nincs is szükség, mivel a λ-kalkulusban csak névtelen függvények definiálhatók.

A funkcionális program végrehajtása a kifejezés értékének meghatározását jelenti. Ez a λ-kalkulusban a program λ-kifejezésének kiértékelését, azaz a legegyszerűbb formára hozását jelenti. A λ-kalkulus a művelet elvégzéséhez átalakítási szabályokat ad, azaz a λ-kalkulus a λ-kifejezések közötti olyan egyenlőségekből áll, amelyek levezethetők a konverziós szabályokból származtatott axiómákból.

A λ-kalkulussal foglalkozunk a könyv I. részében.

Schönfinkel is kidolgozott 1924-ben egy egyszerű elméletet a függvények vizsgálatára. Curry 1930-ban publikálta a kombinátor logika elméletét, ami lényegében a függvények változómentes elmélete volt. Kleene bebizonyította a kombinátor logikának a λ-kalkulussal való ekvivalenciáját, és így kialakult a Turing-kiszámíthatóság, a λ-kalkulus és a kombinátor logika teljes egyenrangúsága.

A kombinátor logikát tanulmányozzuk a II. részben.

A könyv III. részében speciális λ-kalkulusokkal foglalkozunk.

Az 1960-as évek elejére kialakultak azok a módszerek, amelyekkel már könnyen átalakíthatók a funkcionális programok a λ-kalkulus vagy a kombinátor logika kifejezéseire. A funkcionális nyelv implementációjához csak egy redukáló rendszerre volt szükség. Egy ilyen redukáló gépet, a SECD-gépet és a hozzá tartozó λ-kalkulust mutatjuk be a III. rész két fejezetében.

Később, a programnyelvek típusfogalma fejlődésének hatására kialakultak a típusos λ-kalkulus és kombinátor logika rendszerek. Hindley kidolgozta az első *típuslevezetésnek* nevezett módszert, amiből Milner megalkotta az ML funkcionális programnyelv polimorfikus típusrendszerét. Ez volt tulajdonképpen a jelenleg is fejlődő, sok új eredményt adó *típuselmélet* alapja. A III. részben két ilyen típusos λ-kalkulussal is foglalkozunk.