Table 1a — Section 4

Coordination Language	Entities Being Coordinated	Mechanism of Coordination	Medium of Coordination	Semantics/Rules Protocols	Degree of Decoupling	Range of comp languages	Application Domain	Implementation Status
Linda	active tuples	tuple exchange	shared tuple space	associative pattern matching	coordination primitives	wide range of comput. models	data parallel programs	different robust implementations
Bauhaus Linda	active tuples	multisets	hierarchies of tuple spaces	set inclusion	coordination primitives	wide range of comput. models	groupware	Unix-based prototype
Bonita	processes	single or group tuple handling	multiple tuple spaces	associative pattern matching	coordination primitives	wide range of comput. models	data “batch parallel” programs	PVM-based using the Linda-kermel
Law-Governed Linda	processes supervised by “law enforcers”	tuples enhanced with control info	logically structured tuple space	laws defining acce-ptable tuple access	coordination rules written in Prolog	many shared-dataspace models	open and secured (distributed) systems	needs h/w support to enforce the laws
Objective Linda	objects	ADTs and logicals (object refs)	multiset, hierarchies of object spaces	type interfaces	coordination primitives	object-oriented languages	modelling open distributed systems	PVM based prototype
LAURA	servers and clients	exchange of typed forms	shared service space	typed interface de-scription of services	separate service description notation	potentially wide	modelling information systems	prototype based on the ISIS toolkit
Ariadne/HOPLa	hybrid processes	matching of (semi-) structured data	tree-shaped tuple space	flexible records	separate coordination component	potentially wide	collaborative environments	prototype
Sonia	actors (people, s/w tools)	possibly nested tuples	agora (shared tuple space)	typed template associative matching	coordination primitives	Smalltalk-oriented	office automation	prototype
Jada/SHADE	mobile agents	exchange of Java applets	Internet as multipe tuple space	Java/HTML	primitives coupled with Java code	Java	WWW, intranet open systems	prototype
GAMMA	distributed data structures	chemical reactions via fixpoint op	possibly structured multiset (bag)	CHAM	parametric coord. patterns (tropes)	potentially wide	modelling s/w architectures	Connection Machine iPSC2
LO/COOLL	agents as linear multiset objects	interagent broadcast group broadcast	multiset, Forum	Linear Logic	rule based coord. component	logic programming oriented	parallel symbolic computing, DAI	shared memory prototype
MESSENGERS	mobile processes	autonomously executable messages	explicitly partitioned distributed shared mem	intra/inter-object invocation	coordination primitives	potentially wide	mobile computing	Sun based implementation
Synchronisers	objects	constraints on accessing objects	message passing	actor model	constraints specified separately	potentially many OOP languages	object-oriented systems	not known
PCN/Strand	concurrent processes	committed-choice rule selection	shared declarative variables	concurrent logic programming	separate coordination component	many message passing languages	scientific computing	distributed implem. on many platforms
Functional Skeletons	sequential functions	function application and composition	distributed graph structure	string/graph reduction	separate skeleton templates	potentially wide	data parallel programs	various families of skeletons
CoLa	sequential processes	correspondents	hierarchically formed points/ranges of view	message passing	separate coordination component	potentially wide	distributed A.I.	PVM based implementation

Table 1b — Section 4

Coordination Language	Entities Being Coordinated	Mechanism of Coordination	Medium of Coordination	Semantics/Rules Protocols	Degree of Decoupling	Range of comp languages	Application Domain	Implementation Status
Opus	sequential tasks	method invocation	Shared Abstraction	data parallelism	separate coordination component	potentially wide but FORTRAN oriented	scientific computing	prototype on top of Chant
PCL	tools forming family entities	exchanging services via ports	hierarchies of tool family entities	dynamic configuration	separate coordination component (PCL)	potentially wide	modelling system architectures	prototype
Conic	system components	exchanging data via ports	hierarchies of logical nodes	state transitions based on quiescence	semi-separate coord component	PASCAL	configuring distributed systems	Unix/VMS with graphical interface
Darwin/Regis	mostly sequential processes	exchanging data via ports	dynamically evolving component graphs	dynamically evolving state transitions	separate fully fledged coord component	C++ oriented	configuring distributed programs	Unix based implementation
Durra	components and resources	events and channel connections	statically defined component graphs	event driven state transitions	separate fully fledged coord component	Ada	rapid prototyping of distributed programs	prototype
CSDL	tools	coordinators speci-fying access rights	group connections	CSCW metaphors	separate fully fledged coord component	potentially wide	cooperative systems	prototype
POLYLITH	software components	event-based triggering	software bus	MILs metaphors	MIL-like specifi-cation syntax	potentially wide	transparent transpor-tation of s/w systems	distributed implementation
Programmer's Playground	devices	implicit communication	discrete/continuous data streams	I/O abstraction	data exchange primitives	potentially wide	distributed multi-media systems	distributed C++ based implementation
RAPIDE	system components	observing/reacting to events	connections between well-defined interfs	poset model	separate coord component	not applicable	prototyping system architectures	prototype
ConCoord	sequential processes	satisfying conditions on domain states	hierarchical config. of event/state domains	pairs of condition-action	separate coordination language (CCL)	potentially wide	distributed/concurrent process oriented algs	prototype on top of Regis
TOOLBUS	software components	exchange of messa-ges and notes	software bus	process oriented T-scripts	separate coordination component	potentially wide	system integration	prototype
MANIFOLD	sequential processes	events and streams	configuration of process networks	event driven state transitions	separate coordination lang (MANIFOLD)	potentially wide	scientific computing s/w architectures	fully implemented on many platforms