atoms

MDPI

Article

Modeling Atomic Structure & Behavior Through Electron Configurations

Stephan Fritzsche [1,2,3,*], Nishita M. Hosea [1,2], Houke Huang [1,2], Tianluo Luo [1,2,4] and Aloka K. Sahoo [1,2]

1 Helmholtz-Institut Jena, Fröbelstieg 3, 07743 Jena, Germany; nishita.hosea@gmail.com (N.M.H.); huanghouke@impcas.ac.cn (H.H.); luotluo@stu.scu.edu.cn (T.L.); a.k.sahoo@hi-jena.gsi.de (A.K.S.)
2 GSI Helmholtzzentrum für Schwerionenforschung, 64291 Darmstadt, Germany
3 Theoretisch-Physikalisches Institut, Friedrich-Schiller-Universität Jena, 07743 Jena, Germany
4 Key Laboratory of Radiation Physics and Technology of Ministry of Education, Institute of Nuclear Science and Technology, Sichuan University, Chengdu 610064, China
* Correspondence: s.fritzsche@gsi.de

Abstract

Electron configurations are known to provide valuable insights into the electronic structure and behavior of atoms. They specify which and how the electronic (sub-) shells are occupied, and are thus an essential ingredient for most atomic observables. When combined with the shell model and the successive filling of shells, these configurations help explain the Periodic Table and much of chemical binding. They also establish a qualitative framework for analyzing excitation, ionization and relaxation processes and may facilitate a wide range of astrophysical and plasma simulations. Here, we review the role of electron configurations for understanding atomic behavior in interactions with particles and radiation. In particular, we identify several central requirements for an efficient treatment of configuration lists and define a *domain-specific* language in order to generate, manipulate and analyze such lists as well as to extract physically relevant information. We also demonstrate the implementation of this language in JAC, the Jena Atomic Calculator. An efficient handling of configurations will refine the coupling of structure codes with the spectral synthesis of plasma radiation, the setup of ionic cascades or even non-LTE plasma simulations. This common framework for dealing with electron configurations therefore improves consistency, reproducibility and scalability of atomic modeling.

Keywords: atomic fine-structure; atomic behavior; domain-specific language; electron configuration; ionic excitation, ionization and relaxation; plasma simulations; shell model; shells and subshells

Check for updates

Academic Editor: Sabyasachi Kar

Received: 23 April 2026
Revised: 4 June 2026
Accepted: 19 June 2026
Published: 23 June 2026

1. Introduction

Electron configurations are frequently displayed in the literature in order to specify which orbitals are occupied and how many electrons reside in each atomic (sub-) shell. These configurations provide a rather compact notation of the electronic structure of atoms and ions and, hence, help explain most of their observables, at least qualitatively [1,2]. They also help classify atomic excitation, ionization and relaxation processes by means of well-defined changes in the occupation of shells. In the modeling of atomic cascades [3], moreover, electron configurations permit a natural decomposition of the atomic computations into tractable steps. Therefore, extended lists of configurations form quite a natural bridge between elaborate atomic structure calculations as well as atomistic plasma simulations of various kinds, whenever level-resolved data are needed for different charge states.

Indeed, electron configurations underpin the atomic shell model that explains the quantized nature of matter at atomic scales. In its simplest form, perhaps, this model readily reveals the structure of hydrogen-like ions and their spectral line series, such as Lyman and Balmer. Today, the atomic shell model forms the foundation of modern atomic-structure theory, and also explains the Periodic Table as well as much of the chemistry of elements. For example, this model predicts that elements from the same group in this table have a homologous valence-shell structure and, hence, a very similar chemical behavior. In addition, the shell model supports effective theories in modern quantum optics [4], plasma physics [5] or semiconductor design [6].

The central motivation for using multiple configurations is electron correlation; the mutual Coulomb repulsion between electrons causes their motions to be correlated in a manner that a single-configuration, independent-particle description cannot capture. Configuration-interaction (CI) expansions can account for these many-electron contributions, if the wave function is taken as a linear combination of configuration-state functions (CSF). Several CI-based strategies have been worked out through the past decades: the complete active space method includes all excitations within a chosen set of active orbitals; many-body perturbation theory and coupled-cluster approaches treat correlation order by order; and the restricted active space (RAS) method partitions the orbital space into subsets with explicit constraints in the occupation of shells, making it computationally tractable even for open-shell atoms and heavy ions [7,8].

Historically, one or just a few electron configurations have been utilized to classify and compute atomic level structures. Configuration lists of increasing size have later been employed especially in various RAS computations in order to realize accurate levels and excitation energies [7,9]. These RAS computations are particularly useful for medium- and high-Z atoms where full configuration interaction is too costly [8]. Different radiative properties of atoms and ions, such as electric-dipole (E1), magnetic-dipole (M1) and further transition matrix elements, Einstein A coefficients and oscillator strengths have been calculated reliably within RAS frameworks [10,11]. Today, this framework also allows elaborate computations of hyperfine constants [12], isotope shifts [13], and other nuclear-dependent observables [14], including relativity, leading quantum-electrodynamical and correlation contributions. In practice, therefore, RAS calculations usually help balance accuracy, computational cost and transparency in atomic-structure and plasma studies [15–17].

Besides detailed level calculations, electron configurations support modeling the excitation and decay (dynamics) of atoms and ions through multiple charge states [18]. In these "chains" of atomic de-excitation processes, electron configurations easily map the possible transitions between ionic charge states and, hence, indicate where vacancies or holes occur, until the ground state (configuration) is reached. For an efficient treatment of such excitation and decay cascades, however, the implementation of a proper domain-specific language (DSL) is required to deal with the extended size and variation of the underlying lists of configurations. Basic features of such a DSL should include the sorting and writing of configurations in an accessible form, identifying closed and open subshells or to extract details on the occupation or coupling of shells, just on demand. Additional features refer to the generation of excited or ionized configurations, relative to some given reference configuration(s) or to the identification of allowed transitions, based on selection rules as well as to multiple display modes for a transparent work with such electron configurations.

Here, we review the role of electron configurations in atomic structure and collision theory, where they have been widely used in standard atomic-structure programs. Often, the occupation of these configurations are encoded as ordered lists of occupied orbitals with well-defined principal and angular-momentum quantum numbers. Until the present, however, most codes treat configurations just as *generators* of CSF, the basis states of the many-electron Hilbert space, but not as independent physical objects. Only a few modern implementations make these configurations explicit enough to control the virtual excitation of configurations or the growths of active spaces systematically. This is quite in contrast to the needs of many applications that rely on long lists of configurations for some detailed atomic and plasma modeling. To handle these demands efficiently, we have developed a reduced DSL for configuration lists. This DSL has been implemented in JAC, the Jena Atomic Calculator [19], and now facilitates (1) reference-based generation of configurations, (2) extraction of electron numbers, open shells, term symbols, etc., as well as (3) multiple display modes, including the compact notation of the electronic core or mean occupation.

In the next section, we first recall the potential utility of electron configurations in different research fields. For this purpose, we briefly recall the use of configurations and how they describe atomic behavior across different situations; cf. Figure 1. Applications include the transformation of atomic states between coupling schemes and the modeling of atomic cascades. Configurations also provide a natural bridge to plasma modeling and help reduce the complexity in physics simulations. Section 3 later discusses the implementation within the JAC toolbox, including the distinction of themes and the explicit data handling. In particular, we show how configurations can be generated or manipulated, and how relevant information can be extracted efficiently. Finally, Section 4 presents a summary and outlook for future applications and developments.

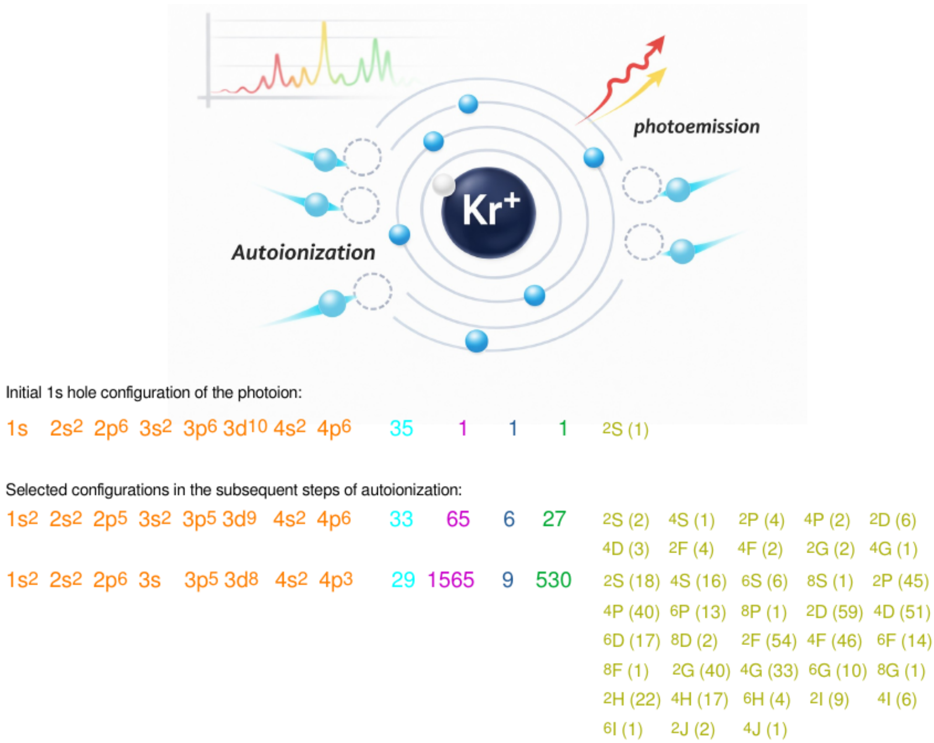

Figure 1. Selected electron configurations in the atomic decay cascade that follows the $1s$ photoionization of Kr^+ $(1s^{-1})$ ions. Apart from the initial $1s$ hole configuration with 35 electrons, two electron configurations are shown that readily occur in the subsequent Auger cascade. These configurations illustrate the rapidly increasing complexity in the fine-structure of ions, if inner-shell excitations are involved. In addition to the occupation of the electron shells (orange, column 1), we also display the numbers of electrons (cyan, column 2), fine-structure levels (pink, column 3), symmetry blocks with well-defined total angular momentum J and parity P (blue, column 4), LS terms (green, column 5) and the explicit LS terms along with their degeneracy (sand, column 6). All this information can be readily extracted from JAC by using the functions below. See text for further explanations.

2. Role and Promise of Dealing with Electron Configurations

2.1. Electron Configurations and Equivalent Electrons

Electron configurations readily specify how the electrons are distributed among the atomic (sub-) shells and, hence, are often seen as a blueprint of the atomic shell model. For instance, oxygen (with nuclear charge $Z = 8$) has a ground configuration $1s^2 2s^2 2p^4$, whereas sulphur $(Z = 16)$ from the same group of the Periodic Table possesses the homologous configuration $1s^2 2s^2 2p^6 3s^2 3p^4$. These configurations simply classify the atomic shell structure in terms of equivalent electrons with identical principal and orbital angular-momentum $(n\ell)$ quantum numbers [1]. Figure 1 displays selected electron configurations in the atomic decay cascade that follow the $1s$ photoionization of $\mathrm{Kr}^+ \ (1s^{-1})$ ions. In addition to the occupation of the electron shells, open-shell configurations give rise to the geometrically determined fine-structure of atomic and ionic levels with well-defined term symbols and symmetry properties. This is shown in this figure for two configurations that readily occur in the subsequent Auger cascade. These configurations illustrate the rapidly increasing complexity in the fine-structure of ions, if additional inner-shell excitations get involved.

The geometrically fixed coupling of the open shells usually ascertains the behavior of most ions, atoms and (small) molecules. If inner-shell excitations occur, these electron configurations enable one to reveal the excitation and decay dynamics by just identifying the principal decay paths. Apart from their explicit fine-structure, these electron configurations also help classify the relevant transitions or to recognize and model weak processes with two or more electrons in the continuum [20–22]. Table 1 summarizes several further research topics that centrally rely on the concept of electron configurations.

Table 1. Research topics that depend on and heavily use electron configurations and, hence, require their efficient handling and manipulation.

Topic & Brief Explanation
Term and level structure of atoms and ions: The knowledge of the term and fine-structure of the low-lying levels has been found crucial for interpreting atomic spectra of different kind and origin, or for validating atomic data, such as compiled in the NIST database [23]. Electron configurations reveal these structures and help identify missing levels or inconsistencies in spectroscopic observations.
Restricted active-space (RAS) atomic computations: Lists of configurations are often generated from given reference configurations by allowing virtual excitations of electrons to predefined active shells. Layer-based RAS schemes then control these excitations by restricting the number of electrons that may occupy different orbital layers or by employing selection rules for parity and angular momentum. This procedure enables one to select compact and physically relevant configuration spaces for accurate atomic-structure calculations.
Transformation between coupling schemes: Transformations between coupling schemes allow the representation of atomic levels in terms of different bases, such as LSJ or jjJ-coupled CSF. All unitary transformations always act upon subspaces as defined by shell structure of the underlying configurations [24,25].
Relaxation of inner-shell holes: Ions with inner-shell holes arise when atoms absorb energetic photons or collide with fast particles [26,27]. The relaxation of these unstable hole states proceeds through characteristic x-ray emission and non-radiative Auger or Coster-Kronig decays. Identifying the dominant decay paths clarifies which configurations and charge states appear during the cascade and may help understand radiation damage. This relaxation is closely related also to the setup and analysis of cascade models, if the relaxation proceeds via several decay steps.
Relevance of weak, second-order processes: Selection rules help identify configurations that mediate weak or two-electron processes, such as the collective Auger decay [28]. These configurations classify the allowed intermediate paths between initial and final states and are particularly important for modeling all processes with two or more electrons in the continuum.
Ions in plasma: In most plasma simulations, the electronic structure is simplified by grouping excited states into *superlevels*, or if just a few bound levels are considered explicitly [29]. This simplification is useful and necessary because most plasma properties depend primarily on averaged populations and rate coefficients, rather than on individual fine-structure levels.
Modeling of astrophysical observations: In non-LTE simulations, the reduced level structures can often be entirely based on electron configurations. The configuration still preserve the overall opacity and emissivity, while reducing the number of transitions [30,31]. Electron configurations remain essential also for identifying the allowed transitions.
Generation and maintenance of atomic databases: Electron configurations serve typically as the key principle for generating and structuring atomic databases. They allow the systematic labeling of levels, terms and transitions, and thus help extract and compare the data in different applications.

2.2. Transformation and Analysis of Coupling Schemes

Perhaps the most straightforward use of electron configurations refers to RAS computations that are usually performed for improving the ionic level structure systematically. In non-relativistic theory, these configurations define how electron angular momenta couple to form terms, levels or individual states. This simple categorization of energy levels by their (allowed) L, S and J values has been found crucial for understanding atomic spectra by means of level structures and selection rules. Many atomic calculations therefore begin from a reference set of configurations to generate relevant subspaces of the many-electron Hilbert space. By using different coupling schemes, the interpretation and synthesis of these spectra can be simplified by reducing the number of relevant terms in the computations [1].

Different coupling schemes can be chosen according to the dominant interactions in the atom or ion. LS (or Russell–Saunders) coupling applies, when the Coulomb repulsion among the electrons exceeds the spin–orbit interaction, as it is usually the case in light atoms. In this coupling scheme, the individual orbital angular momenta L_s of each shell of equivalent electrons, e.g., the so-called symmetry-adopted shell states, are first combined to a total L, similar as for the shell spins S_s to a total S, and before they are coupled to each other to form the total angular momentum J [1]. In heavier atoms, spin–orbit interactions often dominate and then favor jj-coupling, where the total (sub-) shell angular momenta J_s of the equivalent electrons are directly combined to a total J of the given fine-structure level [2].

Not much needs to be said about all the quantum numbers above that characterize the different coupling schemes in the notation (and construction) of atomic and ionic levels. These quantum numbers are well conserved only for nearly pure (many-electron) states, but which rarely occur in nature at all. Although the transformation between orthogonal bases of different coupling schemes is widely known to be achieved by a "unitary matrix", the explicit evaluation of such transformation matrices still remains a challenge for general shell structures and coupling schemes. Despite their known limitations, therefore, LS- and jj-coupling are the most commonly applied schemes in modern atomic theory.

In practice, however, many atoms and ions are not adequately described by either pure scheme, and then require to utilize hybrid coupling hierarchies. In the jK (or $j_1\ell$) scheme, the angular momentum of an inner electron first combines with the orbital angular momentum of an outer electron to form an intermediate quantum number K, which is then coupled to the outer electron's spin to give the total J. The analogous LK scheme couples the total orbital angular momentum L of a group of inner-electrons to the angular momentum of the valence electron. Another widely used variant is LSj coupling, in which the inner core is treated in LS coupling while one or two outer electrons are added in jj-fashion. These and other reordered schemes are frequently encountered in spectroscopic analyses of medium and heavy elements; a comprehensive account is given in [32].

The coupling of the angular momenta can follow also genealogical (or parentage) coupling schemes, which provide a systematic way to build many-electron states. These schemes couple the angular momenta for all shells in a predefined order. Obviously, the chosen coupling sequence determines the set of intermediate quantum numbers that label each step of the construction and result in unitarily equivalent but differently labelled bases. The use of genealogical schemes underpins the coefficients of fractional parentage (CFP), which relate the wave function of an N-electron system to its $(N-1)$-electron parent states and are central to the evaluation of matrix elements over equivalent electrons [1]. Understanding these genealogical hierarchies clarifies why a complete treatment of coupling schemes must go beyond the standard summation order, typically from the inner to the outer shells.

In spite of the similar notation of subshells in both schemes, the anti-symmetrized shell states in *LS*- and *jj*-coupling refer to different basis sets in the irreducible representations of the SO_3 rotation group for a given total angular momentum *J*. This distinction is readily seen already from the splitting of all non-relativistic shells (apart from *ns* shells) into two subshells. The lack of providing a fast and proper spectroscopic notation in various (semi-) relativistic computations has hampered the spectroscopic level classification of medium and heavy elements as well as the interpretation and analysis of inner-shell processes [33,34]. Here, an efficient treatment of electron configurations may help advance JAC to incorporate additional coupling schemes and to enhance the level classification of open-shell atoms and ions in the analysis of inner-shell processes for medium and heavy elements [25].

2.3. Modeling Atomic Cascades

Atomic cascades are ubiquitous in nature and arise whenever an inner-shell electron is excited or ionized, quite independent of how the initial hole has been formed before. While the photoemission, associated with these cascades, can be observed quite readily, the autoionization often dominate the relaxation and must be taken into account in the modeling of all cascades [3]. The photon and electron peaks in the observed spectra first of all reflect the underlying electron configurations, because the fine-structure of the open-shell configurations can hardly be resolved in practice; cf. Figure 1. Beyond the study of inner-shell phenomena, cascades play a crucial role in plasma modeling where they are known to determine the ionization balance and radiation losses. The cascades from inner-shell excited ions also affect the (X-ray) line emission in astrophysical spectra [35] and the radiation damage of biological and material systems [36,37], though often with special emphasis upon the Auger electrons in the latter application.

Indeed, electron configurations provide the proper "building blocks" for addressing atomic cascades since they readily indicate (a) the orbitals to be vacated or filled as well as (b) the transitions that become allowed during the relaxation process. In this stabilization, the cascade is mainly driven by the inner-shell holes, which are filled by electrons from higher shells. The ordering of shells in the underlying configuration already reflect the sequence in which they are likely filled in course of the first, second or any later step of the relaxation. The consequent use of electron configurations allows a systematic description of the essential spectroscopic features and enables one to divide the overall cascade into well-defined computational steps or multiple decay paths, which can be characterized by branching ratios.

Systematically generated lists of electron configurations, based on the photoemission and autoionization of ions, make atomic-cascade modeling feasible, including the initial excitation processes. These lists therefore provide a structured framework for dealing with the excitation and decay of atoms. The concept of atomic cascades can be further expanded in order to imitate atomic behavior under quite different circumstances. This concept has been found useful especially for modeling photoabsorption, generating synthetic spectra and for computing a broad class of plasma rate coefficients [38]. Different cascade schemes have been compiled and discussed in JAC in order to predict cross sections, rate coefficients, electron and photon spectra as well as ion distributions, related to the recombination or (auto-)ionization of ions. Quantitative analysis of cascades often failed in the past simply because of their inherent complexity. By using electron configurations and various predefined cascade *schemes*, an extended framework has been introduced and made such computations feasible. These recent developments represent an important advance in modeling atomic cascades and will facilitate the interpretation of forthcoming observations [3,39].

2.4. A Domain-Specific Language for Dealing with Electron Configurations

The diverse research topics in Table 1 suggest and motivate the development of a DSL for dealing with electron configurations. Such a DSL should cluster features, such as (1) the generation and manipulation of configuration lists, (2) extraction of information for groups of configurations, or (3) the display of electron configurations in a compact or graphical format [40,41]. In contrast to general-purpose languages, a DSL is typically restricted in scope and optimized for a narrowly defined domain, and it is built upon established conventions. The rather precise definition of a DSL also ensures minimal ambiguity as well as high readability. In practice, DSLs are often efficient for expert users but may appear less transparent to incomers [42,43].

To deal efficiently with electron configurations, such a domain-specific language must provide a compact and distinct notation for shells, subshells, occupancies and coupling schemes. Moreover, it might provide systematic rules to describe and deal with parameterized configuration sets. A DSL on electron configurations should also support the extraction of term symbols, multiplicities, the (numbers of) closed or open shells or further details about the terms or fine-structure levels as associated with one or a list of configurations. Table 2 summarizes several central features that appear necessary for an efficient handling of configurations in practice.

Table 2. Important features for handling individual and multiple electron configurations. Each feature has a specific scope and requires basic familiarity with the notion of configurations but enhances clarity and readability in applications. Tables 3–5 below illustrate how these features are implemented within the JAC framework.

Feature & Brief Explanation
Extract information from a configuration: Often, a direct and simple access is needed to all properties that are encoded in a configuration, including the shell occupation and subshell structure. Other information comprises parity, spin multiplicity and possible (total) angular momenta of the levels. A fast access enables a rapid classification of atomic states and transitions.
Derive details about configurations: Atomic levels need typically to be analyzed with respect to their leading configuration or (sub-) dominant configuration components. Such information help clarify configuration mixing and to interpret spectroscopic data.
Generation of configuration list: The systematic creation of excited, ionized or recombined configurations, relative to some chosen reference configuration, is crucial for most applications. These lists define the accessible configuration space for atomic-structure or cascade calculations.
Condense configuration lists under given criteria: Configurations from large configuration sets need to be divided according to well-defined criteria. Typical reductions must enable the user to distinguish between relativistic and non-relativistic configurations, or to select subsets relevant for specific processes. This feature help keep calculations feasible also for complex atoms and ions.
Display (lists of) configurations: A clear and visual presentation of single configurations or extended configuration lists is mandatory. A simple display allows the inspection, comparison and validation of configurations in the input for atomic-structure or plasma applications. It also supports the analysis, how individual configurations affect the behavior of atoms and ions.

Table 3. Functions of the JAC toolbox that support the generation, manipulation and display of individual or lists of configurations. All these functions are based on predefined *themes* and are implemented in the modules `Basics`. Here, we just provide a brief explanation of these functions, whereas further details are given in the documentation of JAC [44] or from Julia's help facilities [45]. See Tables 4 and 5 for supported themes in the given implementation.

Function & Brief Explanation
`displayConfiguration`: to display a given list of configurations in a compact format; additional details about these configurations can be extracted and printed as well. The optional argument `longForm::Bool` determines whether all shells are displayed explicitly, or only the valence-shells are listed as typical for medium and heavy elements.
`displayConfiguration`: to display the same but for a list of configurations. An optional argument `details::String` enables the user to provide further information to the printout.

Table 3. *Cont.*

Function & Brief Explanation
`extractConfiguration`: to extract a single—relativistic or non-relativistic—configuration due to some given theme; cf. Table 4.
`extractConfigurations`: to extract one or several configurations from a basis, level or given multiplet due to some suitable `theme`.
`extractFromConfiguration`: to extract detailed information from or about a given configuration due to some suitable theme.
`extractFromConfigurations`: to extract the same but from a list of configurations due to some suitable theme.
`generateConfigurations`: to generate a list of electron configurations for a given set of (reference) configurations as well as for some suitable theme.

Table 4. Themes to extract individual configurations or information about such configurations from basis functions, the representation of levels and multiplets, or from lists of configurations. Each `theme` is a (concrete) subtype of the abstract data type `Basics.AbstractConfigurationTheme` and may come with own subfield data. All these themes are defined in the module `Basics`. Apart from a brief explanation in this table, further details can be found in the documentation of JAC [44] or from Julia's help facilities [45].

Theme & Brief Explanation
`ByMultipoles()`: ... to extract configurations that are connected to some reference configuration in terms of the well-known multipole selection rules.
`ByParity(P::Parity)`: ... to extract all configurations of given parity P.
`ClosedCore()`: ... to extract the closed core from one or several configurations.
`ClosedShells()`: ... to extract the shells that are filled in one or several configurations.
`ClosedSubshells()`: ... to extract the same but applied to explicit subshells.
`FromBasis()`: ... to extract all configuration that contribute to a given many-electron `basis::Basis`.
`FromNonrelativisticBasis()`: ... to extract all configurations from a non-relativistic `basis::BasisNR`.
`GeneralizedConfigurations()`: ... to extract the generalized configuration of a given set of configurations as often applied in plasma physics.
`GetParity()`: ... to determine the parity of either a relativistic or non-relativistic configuration.
`IsOccupied()`: ... to determine whether a shell or subshell is occupied in one or several configuration(s).
`LeadingConfiguration()`: ... to extract the leading configuration in the representation of a given `level::Level`.
`LeadingConfigurationR()`: ... to extract the same but in terms of leading relativistic configuration.
`Multiplicity()`: ... to determine the multiplicity of a configuration.
`NumberOfElectrons()`: ... to determine the numbers of electrons of one or several configurations.
`OccupationDifference()`: ... to extract the differences in the occupation numbers between two given—either relativistic or non-relativistic—configurations.
`OpenShells()`: ... to return all open shells of one or several configurations.
`OpenSubshells()`: ... to extract the same but for open subshells.
`TotalAM()`: ... to determine the total angular momenta J that are associated with the fine-structure of the given configuration.
`ValenceOccupation()`: ... to extract the occupation of all valence shells in a given configurations with regard to some core configuration.

Table 5. Themes which are frequently used for the generation of configuration lists by starting from one or several reference configurations. Apart from the addition, excitation or removal of electrons, these themes refer to selected processes and, hence, to the setup of cascade models of different depth and complexity. All other details are analogue to Table 4.

Theme & Brief Explanation
`AddElectrons()`: ... to add one or several electrons in specified shells to the given (reference) configurations.
`ExciteElectrons()`: ... to excite one or several electrons w.r.t. the given (reference) configurations.
`RemoveElectrons()`: ... to remove one or several electrons in specified shells from the given (reference) configurations.
`ForAutoIonization()`: ... to generate all those configurations that are related to given (reference) configurations by autoionization, i.e. by a single de-excitation *and* the ejection of an electron. However, no checks are made that such an autoionization is energetically feasible.

Table 5. *Cont.*

Theme & Brief Explanation
`ForDielectronicCapture()`: …to do the same but for the dielectronic capture into the given (reference) configuration; this includes the excitation of one electron and capture of another electron into one of the specified shells.
`ForElectronCapture()`: …to do the same but for the capture of an electron into specified shells and with regard to one or several (reference) configurations.
`ForHollowIons()`: …to do the same but for the multiple capture into (high-n) shells and with regard to one or several (reference) configuration.
`ForPhotoEmission()`: …to do the same but for photoemission and related to one or several (reference) configurations.
`ForPhotoIonization()`: …to do the same but for photoionization, i.e. the removal of an electron, and related to one or several (reference) configurations.
`ForPhotoRecombination()`: …to do the same but for the capture of an electron, e.g. the (radiative) recombination of one electron into specified shells, and related to one or several (reference) configurations.
`ForStepwiseDecay()`: …to do the same but for the release of $\leq n_{\text{total}}$ electrons due to the stepwise photoemission and autoionization, and related to one or several (reference) configurations.

2.5. From Atomic Configurations to Astro and Plasma Applications

Astrophysical spectra typically arise from transitions between atomic energy levels in dilute plasmas. In these spectra, the observed line intensities depend not only on the radiative rates or branching ratios but also on the level population and the rates of various collisional processes. The limited spectral resolution of observations often hampers the identification of fine-structure in many measurements [46]. In the analysis of astrophysical spectra, therefore, levels are often combined into effective states such that the total population is preserved. Electron configurations govern the grouping of levels due to their symmetry or sharing common decay channels. This grouping should still capture the population kinetics, radiative transfer and the ratio of diagnostic lines. Such a reduction also establishes a practical bridge between atomic detail, the synthesis of useful spectra and the computational costs [47,48].

In plasma physics, (so-called) *superconfigurations* have been applied to group ordinary configurations with similar shell occupations into some coarse-grained entities [49]. These superconfigurations may link the electronic structure of atoms and ions to the modeling of plasma at finite temperature and density [50]. In most plasma simulations, only a limited set of bound levels is retained to reduce either, again, the costs or to just model a modest number of charge states by rate equations.

Finally, the configuration concept can be combined also with the popular average-atom model by replacing the explicit configurations in terms of occupation numbers, averaged over statistical ensembles [51,52]. Configuration-averaged approaches treat shells collectively and neglect any detailed configuration interaction, level mixing or fine-structure contributions. Evidently, this averaging provides efficiency but sacrifices detailed spectroscopic resolution.

3. Implementation and Use of Electron Configurations Within the JAC Toolbox

3.1. A Brief Overview of JAC

JAC is a computational toolbox developed to calculate atomic transition amplitudes and properties for a wide range of excitation and decay processes [19]. It is applicable for atoms and ions with open shells and has been designed to provide a general, easy-to-use platform for the atomic physics community by integrating different atomic processes within a single program. Key design principles of the JAC toolbox include: (1) a clear user interface to express the input in quite natural form, (2) the support of atoms and ions with general shell structure and (3) a transparent data flow, independent of some particular application. Since JAC's initial release in 2019, we have steadily expanded the number of atomic properties and processes it can handle. The toolbox ensures good self-consistency in the generated data and allows reproducible and scalable modeling of atoms and ions with complex shells.

JAC differs from other atomic codes by providing a simplified handling and control of atomic processes, approximations and shell structures. Its implementation in the Julia programming language leverages modern features for efficient and flexible computations. The use of JAC requires indeed minimal prior knowledge of the code. Recently, moreover, the JAC toolbox has been opened also towards plasma applications [53], atomic strong-field processes [54] as well as the time-evolution of atomic density matrices, based on Liouville's equation. Users can readily control the computations by using well-designed notations and data structures that improve the reproducibility of the computations, when compared to traditional atomic programs.

In JAC, for example, an electron configuration is encoded by the datatype `Configuration`, which defines a non-relativistic configuration in terms of *shell* labels, such as "1s", "2s", "2p", and the corresponding occupation numbers. Figure 2 illustrates the internal structure and definition of an electron configuration within JAC. The order of shells is irrelevant for this data type, and their number does not reflect the (total) orbital space in the given computations. Different constructors are provided to facilitate the specification of closed cores, such as "[He]" or "[Ne]" or even of bare ions.

```
struct Basics.Configuration ... defines a struct for a non-relativistic electron configuration that
is specified by its shell notations (such as "1s", "2s", "2p", ....) as well as the corresponding
occupation numbers (>= 0). An electron configuration is independent of the sequence of shells,
and a zero occupation is assumed for all shells that do not appear as key in the shells
(dictionary).

+ shells         ::Dict{Shell,Int64}    ... Dictionary that maps shells to their occupation.
+ NoElectrons    ::Int64                ... Number of electrons.


Basics.Configuration(NoElectrons::Int64) ... constructor for a given number of electrons that are
just "filled" according to the standard order of the shells and subshells.
A conf::Configuration is returned.


Basics.Configuration(sa::String) ... constructor for a given configuration string, such as
"[He]", "[Ne]", "[Ne] 3s 3p^6" or "1s 2p^6 3s^2 3p". One can also use "1s^0" in order to represent
a bare-ion.
```

Figure 2. Definition of the data structure `Basics.Configuration` to enter and deal with electron configurations within the JAC toolbox.

In JAC, a still reduced DSL for electron configurations provides a compact and unambiguous notation for dealing with the underlying shells, subshells, occupancies or coupling schemes. It implements systematic rules to display, manipulate and extract information from configurations. Implicit defaults and inheritance are used, for instance, to omit closed shells or to define parameterized configuration sets. Key functionalities of this DSL include the reference-based generation of configurations through excitation, ionization or hole creation; extraction of information, such as the numbers of electrons, open shells or term symbols; and various display modes for a compact notation.

With the present implementation of a DSL for electron configurations, we here provide tools for a compact and unambiguous notation for handling all these details. Implicit defaults and inheritance are used, for instance, to omit closed shells from the specification or to define whole lists of configurations. Table 3 summarizes the JAC functions that facilitate these manipulations. The function `Basics.extractFromConfiguration` allows a theme-based extraction of rather specific information about one or several configurations.

Other, more advanced features enable the analysis, computation and ordering of physical processes, including spectra, cascades, ionization and chemical bonding.

3.2. Different Themes for Dealing with Electron Configurations

In a DSL, different themes distinguish recurring concepts and operations that are needed to provide useful functionality within the domain. These themes organize the language semantically and enables us to implement, document and extend the code in a systematic manner [55]. Many themes are implemented with additional parameters to control their behavior. Table 4, for instance, displays the themes in JAC that help extract individual configurations or information about them from basis functions, the representation of levels or multiplets, or simply from lists of configurations. These `theme`'s are concrete subtypes of the abstract data type `Basics.AbstractConfigurationTheme` and are all defined in the module `Basics`. The use of these themes (data structures) improves readability, reduces errors and facilitates efficient processing of configuration data [56].

The implementation qualifies as an embedded (or internal) DSL in the sense of Hudak [40] and Kosar et al. [55], both of whom distinguish between external DSL—which require their own parser and grammar—and internal DSL that are embedded in a host language and exploit its own type system, dispatch mechanism and evaluation model to define a domain-specific vocabulary and semantics. In JAC, the abstract type `Basics.AbstractConfigurationTheme` and its concrete subtypes (`ForAutoIonization()`, `ClosedShells()`, `FineStructure()`, ...) constitute the domain vocabulary: these identifiers receive their meaning exclusively within the context of atomic-physics configurations and have no utility in general-purpose computation. Julia's multiple-dispatch mechanism then acts as the semantic resolution layer that maps each (*function*, *theme*) pair upon the appropriate domain operation. This role is quite analog to the grammar rules in an external DSL. The scope is deliberately restricted according to Hudak [40], in the sense that the language cannot express arbitrary computation over configurations but only those operations listed in Tables 3–6. This design closely parallels also the embedded DSL pattern used, for example, by Devito for finite-difference geophysical computations [41].

Table 6. Themes which are used in the classification and display of electron configurations. All other details are analogue to Table 4.

Theme & Brief Explanation
`FineStructure()`: ... to display the fine-structure levels of a configuration in terms of the total J and their degeneracy but without the computation of energies.
`FineStructureLS()`: ... to display the total LS terms and their degeneracy but (again) without the computation of energies.
`GroundConfiguration()`: ... to generate the ground configuration of an ion with nuclear charge Z and for just a given number of electrons.
`HyperfineStructure()`: ... to display the hyperfine levels of the configuration for a given nuclear spin I in terms of the total F and their degeneracy but without the computation of energies.
`MeanConfiguration()`: ... to generate the mean configuration for a given set of configurations, i.e. a configuration with mean occupation numbers.
`RelativisticConfigurations()`: ... to refer to the use and analysis of relativistic configurations.
`SuperConfiguration()` ... to generate all configurations that are described by some given superconfiguration (not yet well supported).

A number of themes, compiled in Table 5, help generate lists of configurations for selected atomic processes by starting from one or several reference configurations. These lists can be used to design and construct cascade models of different depth and complexity.

Further themes are provided to directly add or remove electrons from certain shells in the given reference configurations. Table 6, finally, lists themes which can be employed in the classification and display of electron configurations. In the following sections, we shall illustrate how these themes can be utilized in practice.

3.3. Generation and Extraction of Electron Configurations in JAC

Many interactions of atoms with external particles or fields lead to the excitation, de-excitation, ionization or capture of electrons. If inner-shell electrons become excited, atoms often autoionize under the emission of electrons, to release the excess energy. To understand this relaxation of inner-shell excited atoms and ions, the initial hole (or reference) configuration must be modified stepwise by de-exciting and removing electrons until the system has been stabilized, or if simply a number of electrons were removed from the initial. Although the rules for such modifications are often simple, long lists may arise and are difficult to manage in practice. Table 5 outlines the themes provided for such manipulations. Each theme comprises one or more subfields that define the relevant shells and operations precisely. Therefore, efficient tools are needed to generate, modify and employ these configuration sets transparently.

Let us illustrate the use of these themes in the JAC toolbox. In the K-shell photoionization of krypton, for instance, a $1s$ electron is ionized and the photoion is left in the $\mathrm{Kr}^+ \ (1s^{-1}) \equiv 1s2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6$ configuration as displayed in Figure 1. By starting from this $1s$ hole configuration in our example, we can easily generate all configurations that may arise in a single step of the subsequent autoionization or photoemission. This is simply achieved by the short Julia script:

```
iConfs = [Configuration("1s 2s^2 2p^6 3s^2 3p^6 3d^10 4s^2 4p^6")]
aConfs = Basics.generateConfigurations(ForAutoIonization(), iConfs)
pConfs = Basics.generateConfigurations(ForPhotoEmission(), iConfs)
```

```
28-element Vector{Configuration}:
 Configuration: 1s^2 2s^0 2p^6 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^1 2p^5 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^1 2p^6 3s^1 3p^6 3d^10 4s^2 4p^6
 :
7-element Vector{Configuration}:
 Configuration: 1s^2 2s^1 2p^6 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^2 2p^5 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^2 2p^6 3s^1 3p^6 3d^10 4s^2 4p^6
 :
```

All proper displacements of electrons are taken into account in these lists and result in 28 autoionizing and 7 other configurations that may follow due to photoemission. Specifically, `ForAutoIonization() generates all single-step Auger and Coster-Kronig channels, i.e., configurations reachable by moving one electron from an occupied shell into a vacancy, while a second electron is removed or placed into the continuum. Multi-electron transitions, such as shake-off, shake-up satellites or collective Auger processes are not included at this level; they require dedicated treatment, for instance, via iterative application of the decay themes or through explicit cascade-block construction [3]`. Similar calls can be made for several other processes, such as the dielectronic capture or photorecombination, if shells are specified for the captured electron. With the theme `ForStepwiseDecay`, moreover, several of these decay steps can be combined and treated together. If we allow two subsequent steps in the autoionization of the $\mathrm{Kr}^+ \ (1s^{-1})$ ions, we shall find, of course, a whole cascade (tree) of configurations which occur in different decay paths.

```
dConfs = Basics.generateConfigurations(ForStepwiseDecay(2), iConfs)
```

```
110-element Vector{Configuration}:
 Configuration: 1s^2 2s^1 2p^6 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^2 2p^5 3s^2 3p^6 3d^10 4s^2 4p^6
 Configuration: 1s^2 2s^2 2p^6 3s^1 3p^6 3d^10 4s^2 4p^6
 :
```

Clearly, this list should now include configurations with 35, 34 and 33 electrons and with single and/or multiple holes in various shells. An analogous call for the ejection of up to four electrons in the cascade results in a list of 428 configurations, which also includes the two decay configurations shown in Figure 1.

3.4. Scrutinize and Display Electron Configurations: Extract Selected Information

Electron configurations represent the shell structure of atoms and ions and, therefore, provide a natural framework in order to report about atomic behavior in different contexts. Apart from their role in atomic-structure and collision theory, or for modeling atomic cascades, configurations form a conceptual bridge to other fields, such as astrophysics, plasma modeling or the analysis and simplification of complex physical systems. In most atomic-structure codes, configurations are encoded as lists of quantum numbers for characterizing the shells and occupation, and with the sole purpose to construct CSF for the setup and diagonalization of the Hamiltonian matrix. Modern frameworks, such as the JAC toolbox, treat configurations as explicit objects in order to control active spaces, excitations or atomic behavior in a more systematic fashion.

Once a configuration is specified, therefore, one should readily be able to extract its key properties as, for instance, the total electron number, the open subshells, possible term symbols due to the coupling of open shells, their spin multiplicity and parity, or several other information. The efficient extraction and handling of such information becomes indeed a central requirement for all applications that are based on configurations.

Among other uses, cascade processes are one of the most frequent applications, in which electron configurations serve as an efficient bookkeeping of different processes, steps and (decay) paths in the overall evolution of the system. An electron configuration primarily determines the shell occupation, indicates the orbitals that need to be *filled* or *emptied*, and identifies the transitions that remain (likely) allowed. Many cascades are driven by the vacancies in inner shells, since electrons from higher shells tend to fill these holes, releasing energy by photon emission or electron ejection. A proper list of configurations therefore reveals which electrons are removed or rearranged in the first, second or some later step of the cascade. In practice, these extended lists of configurations must be generated automatically and can then serve as a starting point to group related configurations into so-called cascade blocks. Indeed, such blocks provide a compact representation of typical decay trees and facilitate the construction of efficient cascade models [22,39].

With this short reminder how configurations appear in applications, we can use the themes in Table 5 to extract information about (lists of) configurations. For example, we can determine the number of electrons, the closed and open shells of the configurations in the list `dConfs`:

```
extractFromConfigurations(NumberOfElectrons(), dConfs)
extractFromConfigurations(ClosedShells(),    dConfs)
extractFromConfigurations(MeanOccupation(), dConfs)
```

```
[35, 35, 35, 35, 35, 35, 35, 34, 34, 34, 34, ..., 33]  # 110 elements in total
Shell[2s, 2p, 3p, 3d, 4s, 3s, 4p, 1s]
Dict{Shell, Float64}(2s => 1.727, 2p => 5.6, 3p => 5.6, 3d => 9.6, 4s => 1.627,
                     3s => 1.627, 4p => 5.6, 1s => 2.0)
```

From the printout of these function calls, we find configurations with 35, 34 and 33 electrons as expected after a photoemission and/or autoionization has happened twice. We also see that all shells 1*s*, 2*s*, ..., 4*p* are filled in at least one configuration, even if the shell order does not follow their natural sequence here. If averaged over all the configurations, finally, only the 1*s* shell is filled, while a fractional occupation applies to all other shells. Here, `MeanOccupation() computes the arithmetic mean of the occupation numbers over all configurations in the list with account of the degeneracy (weighting). For plasma applications where statistically weighted averages are needed--for example, Boltzmann- or level-population-weighted means--these can be alternatively assembled by the user from the individual occupation data returned by NumberOfElectrons() and the relevant level populations.` In addition to these simple requests, other themes can be utilized but may require a valid representation of an atomic level or many-electron basis in order to obtain the desired results. No needs occur to make all these features explicit, whose implementation can be easily understood from Julia's help facilities [45].

Let us explore instead the fine-structure and multiplicity of the second (open-shell) configuration in Figure 1 as it may arise after two autoionizing steps in the decay cascade. For the multiplet of this configuration, we can analyze the fine-structure and hyperfine levels in different but obvious representations. If we run the short script,

```
conf = Configuration("1s^2 2s^2 2p^5 3s^2 3p^6 3d^9 4s^2 4p^6")
Basics.displayConfiguration(stdout, FineStructure(), conf)
Basics.displayConfiguration(stdout, FineStructureLS(), conf)
Basics.displayConfiguration(stdout, HyperfineStructure(AngularJ64(1)), conf)
Basics.displayConfiguration(stdout, HyperfineStructure(AngularJ64(2)), conf)
```

```
Fine-structure of configuration:
  1s^2 2s^2 2p^5 3s^2 3p^6 3d^9 4s^2 4p^6   J's = 0 (1)  1 (3)  2 (4)  3 (3)  4 (1)

LSJ-coupled fine-structure of configuration:
  1s^2 2s^2 2p^5 3s^2 3p^6 3d^9 4s^2 4p^6   ^(2S+1) L's = ^1P (1)  ^3P (1)  ^1D (1)  ^3D (1)  ^1F (1)  ^3F (1)

Hyperfine-structure of configuration for nuclear spin 1:
  1s^2 2s^2 2p^5 3s^2 3p^6 3d^9 4s^2 4p^6   F's = 0 (3)  1 (8)  2 (10)  3 (8)  4 (4)  5 (1)

Hyperfine-structure of configuration for nuclear spin 2:
  1s^2 2s^2 2p^5 3s^2 3p^6 3d^9 4s^2 4p^6   F's = 0 (4)  1 (10)  2 (12)  3 (11)  4 (8)  5 (4)  6 (1)
```

We first obtain the fine- and *LS* term-structure along with the degeneracy of each symmetry. In the last two calls, we have specified in addition also the nuclear spin $I = 1, 2$ to obtain the degeneracy and values of the total angular momentum F. This display of the (hyper-)fine-structure of configurations help understand the role of levels in different spectra. This quite simple example already shows how rapidly the fine-structure increases, if configurations with several open shells occur. Of course, the same or very similar calls have "generated" all numbers in Figure 1. They demonstrate the advantages of a DSL for dealing with electron configurations, as deriving these symmetries by hand is already quite demanding. A fast and reliable access to, and inspection of, these symmetries and quantum numbers can therefore also aid in the interpretation of a wide variety of spectra.

4. Summary and Conclusions

Electron configurations aggregate our knowledge about the (sub-) shell population structure of atoms and ions and, hence, provide first insights into their electronic structure and behavior. Nearly all atomic observables can be understood, at least qualitatively, by starting from this distribution of electrons among the shells. An efficient treatment of electron configurations therefore supports applications from accurate atomic-structure and collision calculations to spectroscopy, to the realm of inner-shell phenomena, and up to astrophysical and plasma modeling, to name just a few.

This work reviews the role of electron configurations with emphasis on atomic processes and interactions. In particular, we have identified central requirements for an efficient use of extended lists of configuration in modern computations. A DSL has been introduced to generate, manipulate and analyze such configurations, and to extract physically relevant information about them. With the implementation of this language into the JAC toolbox, we demonstrate how such a language can be integrated into a unified computational framework. This common framework thereby provides a robust basis for atomic modeling across a range of applications.

Whereas a DSL merely establishes the formal concepts and operations, which are needed to handle such configurations, its practical implementation translates these concepts into executable code with functions, data structures and interfaces. With the current implementation in JAC, we aim to support the generation and manipulation of configuration lists even if neither more advanced correlation models nor any automatic transformation between coupling schemes has yet been incorporated. Nonetheless, this implementation provides a robust and transparent framework for managing large configuration sets. In coming years, we expect to incorporate further atomic processes, improve efficiency and facilitate the coupling with ongoing astrophysical and plasma simulations.

Author Contributions: Methodology, S.F., N.M.H. and A.K.S.; software, S.F., N.M.H. and T.L.; writing—review and editing, S.F., H.H. and A.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All created and analyzed data are directly shown in the text; further data sharing is not applicable to this article.

Acknowledgments: During the preparation of this manuscript, the authors used GPT-5 for the purposes of English improvement and consistency checks. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cowan, R.D. *The Theory of Atomic Structure and Spectra*; University of California Press: Berkeley, CA, USA, 1981.
2. Grant, I.P. *Relativistic Quantum Theory of Atoms and Molecules: Theory and Computation*; Springer: Berlin/Heidelberg, Germany, 2007.
3. Fritzsche, S.; Palmeri, P.; Schippers, S. Atomic cascade computations. *Symmetry* **2021**, *13*, 520. [CrossRef]
4. Celi, A.; Sanpera, A.; Ahufinger, V.; Lewenstein, M. Quantum optics and frontiers of physics: The third quantum revolution. *Phys. Scr.* **2017**, *92*, 013003.
5. Fitzgerald, J.M.; Narang, P.; Craster, R.V.; Maier, S.A.; Giannini, V. Quantum plasmonics. *Proc. IEEE* **2016**, *104*, 2307. [CrossRef]
6. National Academies of Sciences, Engineering, and Medicine. *Manipulating Quantum Systems: An Assessment of Atomic, Molecular, and Optical Physics in the United States*; National Academies Press: Washington, DC, USA, 2020.
7. Fischer, C.F.; Godefroid, M.; Brage, T.; Jönsson, P.; Gaigalas, G. Advanced multiconfiguration methods for complex atoms: I. Energies and wave functions. *J. Phys. B* **2016**, *49*, 182004. [CrossRef]
8. Casanova, D. Efficient implementation of restricted active space configuration interaction with the hole and particle approximation. *J. Comput. Chem.* **2013**, *34*, 720. [CrossRef]
9. Guo, M.; Sørensen, L.K.; Delcey, M.G.; Pinjari, R.V.; Lundberg, M. Simulations of iron K pre-edge X-ray absorption spectra using the restricted active space method. *Phys. Chem. Chem. Phys.* **2016**, *18*, 3250. [CrossRef] [PubMed]
10. Kramida, A. Evaluation of uncertainties in atomic data on spectral lines and transition probabilities. *Eur. Phys. J. D* **2024**, *78*, 36. [CrossRef]
11. Andrews, J.S.; Bondarev, A.I.; Jönsson, P.; Grumer, J.; Raeder, S.; Fritzsche, S.; Bieroń, J. Ab initio multiconfigurational calculations of experimentally significant energy levels and transition rates in Lr I ($Z = 103$). *Phys. Rev. A* **2025**, *112*, 062802. [CrossRef]

12. Yordanov, D.T.; Balabanski, D.L.; Bieron, J.; Bissell, M.L.; Blaum, K.; Budinčevic, I.; Fritzsche, S.; Frömmgen, N.; Georgiev, G.; Geppert, C.; et al. Spins, electromagnetic moments and isomers of $^{107-129}$Cd. *Phys. Rev. Lett.* **2013**, *110*, 192501. [CrossRef] [PubMed]
13. Cheal, B.; Cocolios, T.E.; Fritzsche, S. Laser spectroscopy of radioactive isotopes: Role and limitations of accurate isotope-shift calculations. *Phys. Rev.* **2012**, *A86*, 042501.
14. Lu, B.; Zhang, T.; Chang, H.; Li, J.; Wu, Y.; Wang, J. Reevaluation of the nuclear electric quadrupole moment for ^{87}Sr by hyperfine structures and relativistic atomic theory. *Phys. Rev. A* **2019**, *100*, 012504. [CrossRef]
15. Bieron, J.; Froese Fischer, C.; Fritzsche, S.; Gaigalas, G.; Grant, I.P.; Indelicato, P.; Jönsson, P.; Pyykkö, P. Ab initio MCDHF calculations of electron-nucleus interactions. *Phys. Scr.* **2015**, *90*, 054011.
16. Michel-Dansac, L.; Blaizot, J.; Garel, T.; Verhamme, A.; Kimm, T.; Trebitsch, M. RASCAS: Radiation scattering in astrophysical simulations. *Astron. Astrophys.* **2020**, *635*, A154. [CrossRef]
17. Klene, M.; Robb, M.A.; Blancafort, L.; Frisch, M.J. A new efficient approach to the direct restricted active space self-consistent field method. *J. Chem. Phys.* **2003**, *119*, 713. [CrossRef]
18. Schippers, S.; Stock, S.; Buhr, T.; Perry-Saßmannshausen, A.; Reinwardt, S.; Martins, M.; Müller, A.; Fritzsche, S. Near K-edge photoionization and photoabsorption of singly, doubly, and triply charged silicon ions. *Astrophys. J.* **2022**, *931*, 100. [CrossRef]
19. Fritzsche, S. A fresh computational approach to atomic structures, processes and cascades. *Comp. Phys. Commun.* **2019**, *240*, 1. [CrossRef]
20. Engelns, A.; Klar, H.; Malcherek, A.W. Two-electron atoms in double continua: Asymptotic wavefunctions. *J. Phys. B* **1997**, *30*, L811. [CrossRef]
21. Linusson, P.; Fritzsche, S.; Eland, J.H.D.; Mucke, M.; Feifel, R. Single-photon multiple ionization forming double vacancies in the 2p subshell of argon. *Phys. Rev. A* **2013**, *87*, 043409. [CrossRef]
22. Schippers, S.; Hamann, A.; Perry-Sassmannshausen, A.; Buhr, T.; Müller, A.; Martins, M.; Reinwardt, S.; Trinter, F.; Fritzsche, S. Multiple photodetachment of oxygen anions via K-shell excitation and ionization: Direct double-detachment processes and subsequent deexcitation cascades. *Phys. Rev. A* **2022**, *106*, 013114. [CrossRef]
23. Kelleher, D.E.; Martin, W.C.; Wiese, W.L.; Sugar, J.; Fuhr, J.R.; Olsen, K.; Dalton, G.R. The new NIST atomic spectra database. *Phys. Scr.* **1999**, *T83*, 158. [CrossRef]
24. Gaigalas, G.; Zalandauskas, T.; Fritzsche, S. Spectroscopic LSJ notation for atomic levels as obtained from relativistic calculations. *Comput. Phys. Commun.* **2004**, *157*, 239. [CrossRef]
25. Gaigalas, G. Coupling: The program for searching optimal coupling scheme in atomic theory. *Comput. Phys. Commun.* **2020**, *247*, 106960. [CrossRef]
26. Crasemann, B. Atomic inner-shell threshold excitation phenomena. *J. Phys. Colloq.* **1987**, *48*, C9. [CrossRef]
27. Miron, C.; Morin, P. High-resolution inner-shell photoionization, photoelectron and coincidence spectroscopy. In *Handbook of High-Resolution Spectroscopy*; Wiley: Hoboken, NJ, USA, 2011; Volume 3, p. 1655.
28. Hikosaka, Y.; Fritzsche, S. Amplified collective Auger decay of double inner-shell vacancy in Xe. *Phys. Rev. Lett.* **2025**, *134*, 103001. [CrossRef] [PubMed]
29. Klapisch, M.; Bar-Shalom, A.; Oreg, J.; Colombant, D. Recent developments in atomic physics for the simulation of hot plasmas. *Phys. Plasmas* **2001**, *8*, 1817. [CrossRef]
30. Hubeny, I.; Hubeny, V. Non-LTE models and theoretical spectra of accretion disks in active galactic nuclei. II. Vertical structure of the disk. *Astrophys. J.* **1998**, *505*, 558.
31. Amarsi, A.M.; Grevesse, N.; Grumer, J.; Asplund, M.; Barklem, P.S.; Collet, R. The 3D non-LTE solar nitrogen abundance from atomic lines. *Astron. Astrophys.* **2020**, *636*, A120. [CrossRef]
32. Martin, W.C. Atomic spectroscopy. In *Springer Handbook of Atomic, Molecular, and Optical Physics*; Drake, G.W.F., Ed.; Springer: Cham, Switzerland, 2023; Volume 11, pp. 177–197. [CrossRef] [PubMed]
33. Crasemann, B. *Atomic Inner-Shell Physics*; Springer: Berlin/Heidelberg, Germany, 2013.
34. Müller, A.; Martins, M.; Borovik, A., Jr.; Buhr, T.; Perry-Saßmannshausen, A.; Reinwardt, S.; Ricz, S.; Trinter, F.; Schippers, S.; Fritzsche, S.; et al. The role of L-shell single and double core-hole production and decay in m-fold ($m = 1, 2, \ldots, 6$) photoionization of the Ar^+ ion. *Phys. Rev. A* **2021**, *104*, 033105. [CrossRef]
35. Rosner, R.; Golub, L.; Vaiana, G.S. On stellar X-ray emission. *Annu. Rev. Astron. Astrophys.* **1985**, *23*, 413. [CrossRef]
36. Cosslett, V.E. Radiation damage in the high resolution electron microscopy of biological materials: A review. *J. Microsc.* **1978**, *113*, 113. [CrossRef] [PubMed]
37. Jiang, N. Electron beam damage in oxides: A review. *Rep. Prog. Phys.* **2016**, *79*, 016501. [PubMed]
38. Huang, H.K.; Wen, W.Q.; Huang, Z.K.; Yuan, Y.; Zhang, C.Y.; Si, R.; Wu, S.J.; Chen, C.Y.; Fritzsche, S.; Schippers, S.; et al. Absolute rate coefficients for dielectronic recombination of sodium-like iron ions: Experiment and theory. *Astrophys. J. Suppl. Ser.* **2025**, *278*, 44. [CrossRef]

39. Fritzsche, S.; Sahoo, A.K.; Sharma, L.; Wu, Z.W.; Schippers, S. Merits of atomic cascade computations. *Eur. Phys. J. D* **2024**, *78*, 75. [CrossRef]
40. Hudak, P. Domain-specific languages. In *Handbook of Programming Languages*; Academic Press: London, UK, 1997; Volume 3, p. 39.
41. Louboutin, M.; Lange, M.; Luporini, F.; Kukreja, N.; Witte, P.A.; Herrmann, F.J.; Gorman, G.J. Devito (v3.1.0): An embedded domain-specific language for finite differences and geophysical exploration. *Geosci. Model Dev.* **2019**, *12*, 1165. [CrossRef]
42. Shen, L.; Chen, X.; Liu, R.; Wang, H.; Ji, G. Domain-specific language techniques for visual computing: A comprehensive study. *Arch. Comput. Methods Eng.* **2021**, *28*, 3113.
43. Hellert, T.; Montenegro, J.; Pollastro, A. PhysBERT: A text embedding model for physics scientific literature. *APL Mach. Learn.* **2024**, *2*, 046105. [CrossRef]
44. Fritzsche S. JAC: User Guide, Compendium & Theoretical Background. Available online: https://github.com/OpenJAC/JAC.jl (accessed on 10 April 2026).
45. Julia Online Resource: Julia Comes with a Full-Featured Interactive Command-Line REPL (Read-Eval-Print Loop) That Is Built into the Executable of the Language. Available online: https://docs.julialang.org/ (accessed on 10 April 2026).
46. Hell, N.; Beiersdorfer, P.; Brown, G.V.; Eckart, M.E.; Kelley, R.L.; Kilbourne, C.A.; Wilms, J. Highly charged ions in a new era of high-resolution X-ray astrophysics. *X-Ray Spectrom.* **2020**, *49*, 218.
47. Woo, J.W.; Forrey, R.C.; Cho, K. Astrophysical extended X-ray absorption fine-structure analysis. *Astrophys. J.* **1997**, *477*, 235. [CrossRef] [PubMed]
48. Shen, K.J.; Blondin, S.; Kasen, D.; Dessart, L.; Townsley, D.M.; Boos, S.; Hillier, D.J. Non-local thermodynamic equilibrium radiative transfer simulations of sub-Chandrasekhar-mass white dwarf detonations. *Astrophys. J. Lett.* **2021**, *909*, L18. [CrossRef]
49. Peyrusse, O. On the superconfiguration approach to model NLTE plasma emission. *J. Quant. Spectrosc. Radiat. Transf.* **2001**, *71*, 571. [CrossRef]
50. Bauche, J.; Bauche-Arnoult, C.; Peyrusse, O. Plasma simulations. In *Atomic Properties in Hot Plasmas: From Levels to Superconfigurations*; Springer: Cham, Switzerland, 2015; p. 297.
51. Filevich, J.; Grava, J.; Purvis, M.; Marconi, M.C.; Rocca, J.J.; Nilsen, J.; Johnson, W.R. Prediction and observation of tin and silver plasmas with index of refraction greater than one in the soft X-ray range. *Phys. Rev. E* **2006**, *74*, 016404. [CrossRef] [PubMed]
52. Johnson, W.R.; Nilsen, J. Average-atom treatment of relaxation time in X-ray Thomson scattering from warm dense matter. *Phys. Rev. E* **2016**, *93*, 033205. [CrossRef] [PubMed]
53. Fritzsche, S. Atomic input for modeling ionic mixtures in astrophysical plasma. *Eur. Phys. J. A* **2025**, *61*, 63. [CrossRef]
54. Fritzsche, S.; Böning, B. Strong-field ionization amplitudes for atomic many-electron targets. *Atoms* **2022**, *10*, 70. [CrossRef]
55. Kosar, T.; Bohra, S.; Mernik, M. Domain-specific languages: A systematic mapping study. *Inf. Softw. Technol.* **2016**, *71*, 77. [CrossRef]
56. Strembeck, M.; Zdun, U. An approach for the systematic development of domain-specific languages. *Softw. Pract. Exp.* **2009**, *39*, 1253. [CrossRef]

https://doi.org/10.3390/atoms14070046